FIG. 1

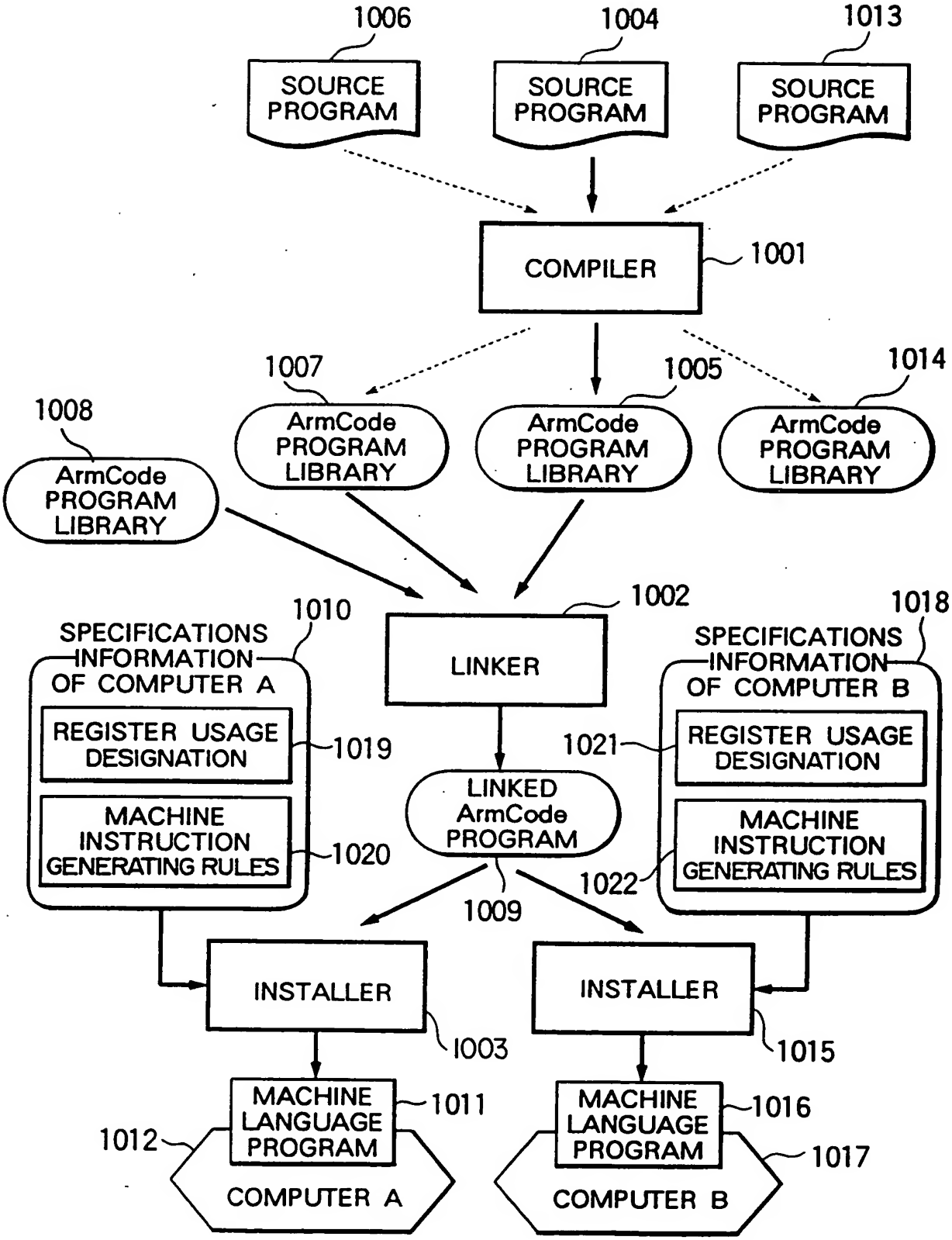

1006
SOURCE PROGRAM
1004
SOURCE PROGRAM
1013
SOURCE PROGRAM
COMPILER
1001
1007
1005
1014
1008
ArmCode PROGRAM LIBRARY
ArmCode PROGRAM LIBRARY
ArmCode PROGRAM LIBRARY
ArmCode PROGRAM LIBRARY
1002
LINKER
1010
SPECIFICATIONS INFORMATION OF COMPUTER A
REGISTER USAGE DESIGNATION
1019
MACHINE INSTRUCTION GENERATING RULES
1020
1018
SPECIFICATIONS INFORMATION OF COMPUTER B
1021
REGISTER USAGE DESIGNATION
MACHINE INSTRUCTION GENERATING RULES
1022
LINKED ArmCode PROGRAM
1009
INSTALLER
1003
INSTALLER
1015
MACHINE LANGUAGE PROGRAM
1011
1012
COMPUTER A
MACHINE LANGUAGE PROGRAM
1016
1017
COMPUTER B

FIG. 2

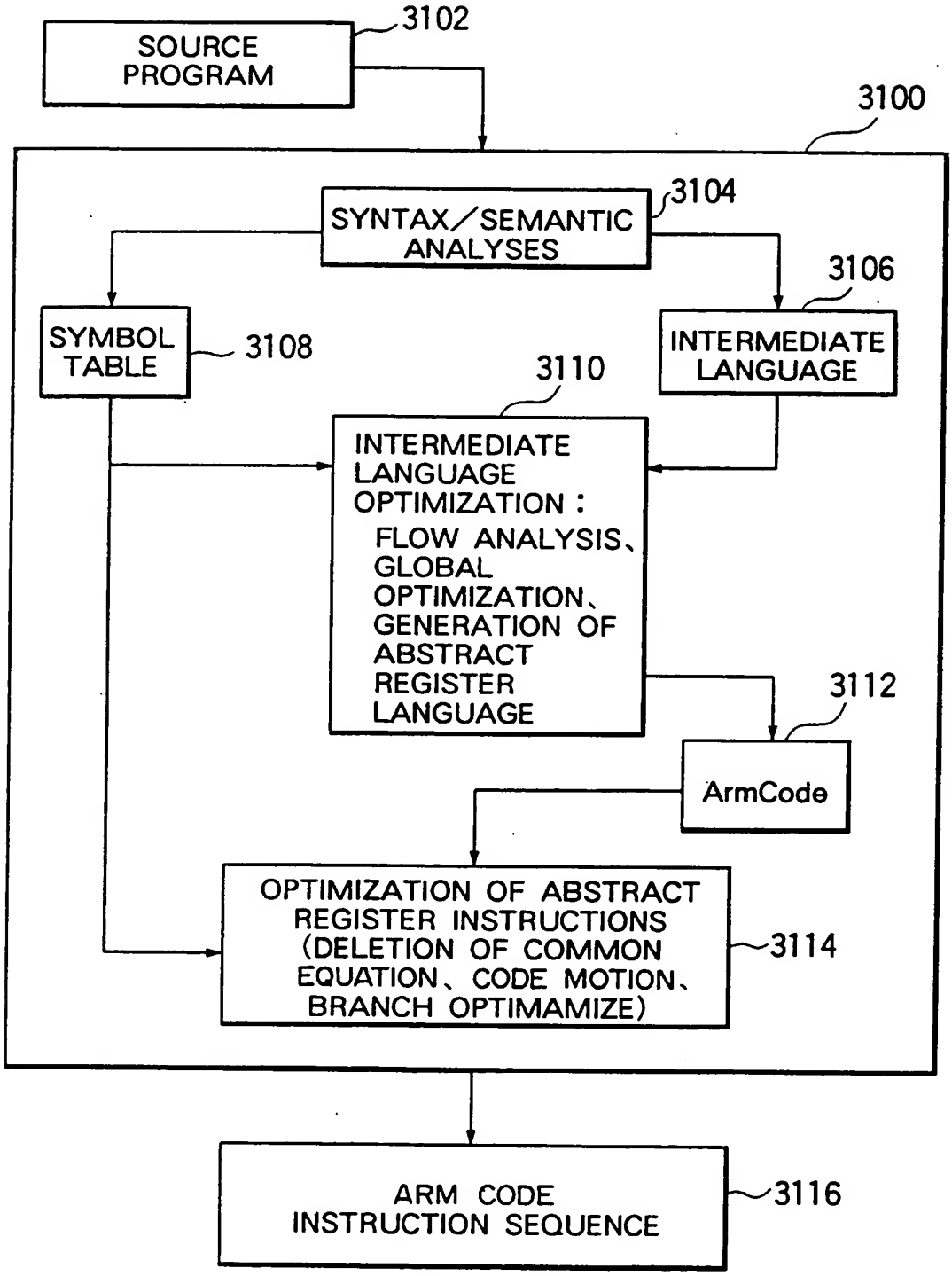

SOURCE PROGRAM
3102
3100
SYNTAX/SEMANTIC ANALYSES
3104
3106
SYMBOL TABLE
3108
INTERMEDIATE LANGUAGE
3110
INTERMEDIATE LANGUAGE OPTIMIZATION : FLOW ANALYSIS、GLOBAL OPTIMIZATION、GENERATION OF ABSTRACT REGISTER LANGUAGE
3112
ArmCode
OPTIMIZATION OF ABSTRACT REGISTER INSTRUCTIONS (DELETION OF COMMON EQUATION、CODE MOTION、BRANCH OPTIMAMIZE)
3114
ARM CODE INSTRUCTION SEQUENCE
3116

FIG. 3

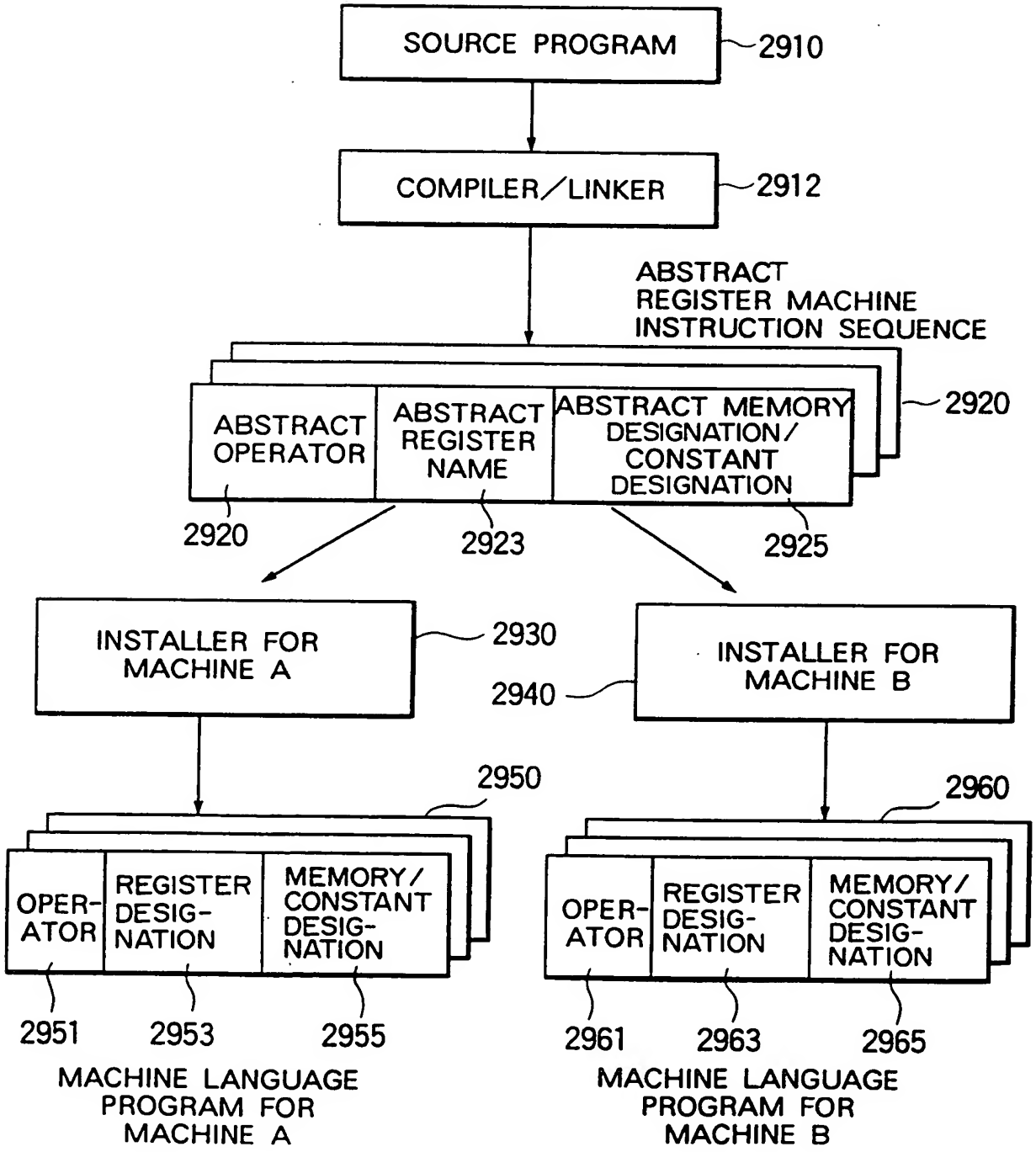

SOURCE PROGRAM
2910
COMPILER/LINKER
2912
ABSTRACT REGISTER MACHINE INSTRUCTION SEQUENCE
ABSTRACT OPERATOR
ABSTRACT REGISTER NAME
ABSTRACT MEMORY DESIGNATION/ CONSTANT DESIGNATION
2920
2920
2923
2925
INSTALLER FOR MACHINE A
2930
2940
INSTALLER FOR MACHINE B
2950
2960
OPER-ATOR
REGISTER DESIG-NATION
MEMORY/ CONSTANT DESIG-NATION
2951
2953
2955
2961
2963
2965
MACHINE LANGUAGE PROGRAM FOR MACHINE A
MACHINE LANGUAGE PROGRAM FOR MACHINE B

FIG. 4

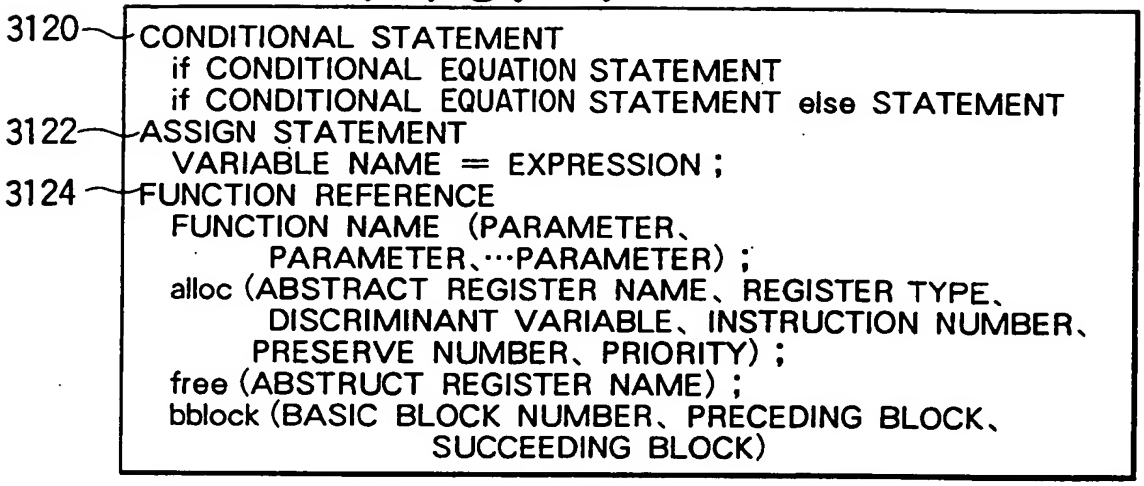
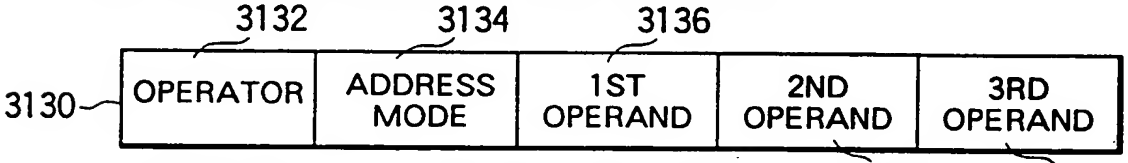


FIG. 5A

INSTRUCTION FORMAT OF EACH ArmCode



OPERATOR	INSTRUCTION NAME
ADDRESS MODE	DISCRIMINATION OF OPERAND FORM
1ST OPERAND	REGISTER、LITERAL OR ArmCode ADDRESS
2ND OPERAND	REGISTER、LITERAL OR ArmCode ADDRESS
3RD OPERAND	MEMORY ADDRESS、DISPLACEMENT、LITERAL、LABEL、ArmCode ADDRESS、ETC

FIG. 5B

OBJECT REPRESENTED BY OPERANDS

3150	REGISTER	ABSTRACT REGISTER NUMBER
3152	MEMORY ADDRESS	ABSOLUTE ADDRESS OR POSITION OF SYMBOL TABLE RERESENTING VARIABLE NAME OR LOGICAL REGISTER NUMBER REPRESENTING INTRA-MEMORY CONSTANT
3154	DISPLACEMENT	NUMERICAL VALUE REPRESENTING DISPLACEMENT RELATIVE TO BASE REGISTER
3158	LITERAL	INTEGER CONSTANT (NUMERICAL VALUES、CHARACTER CODES OR POSITIONS OF CHARACTER STRING TABLE AND COMMENT TABLE AS OBJECT FOR OPERATION)
3160	LABEL	LABEL TABLE NUMBER
3162	ArmCode ADDRESS	POSITION OF ONE INSTRUCTION OF ArmCode

FIG. 6

3170 ADDRESS MODE		3172 OPERAND 1	3174 OPERAND 2	3176 OPERAND 3
RegReg	REGISTER · REGISTER	REGISTER R1	REGISTER R2	
RegM1	REGISTER · MEMORY INDIRECT	REGISTER R1	ADDRESS REGISTER R2	
RegMR	REGISTER · MEMORY RELATIVE	REGISTER R1	BASE REGISTER R2	DISPLACEMENT d
RegMD	REGISTER · MEMORY DIRECT	REGISTER R1		MEMORY ADDRESS m
RegAdr	REGISTER · MEMORY ADDRESS	REGISTER R1	BASE REGISTER R2	DISPLACEMENT d
RegCon	REGISTER · CONSTANT	REGISTER R1		CONSTANT c3
RegRc	REGISTER · REGISTER · CONSTANT	REGISTER R1	REGISTER R2	CONSTANT c3
RegCC	REGISTER · REGISTER · CONSTANT	REGISTER R1	CONSTANT c2	CONSTANT c3
RegRCC	REGISTER · REGISTER · CONSTANT	REGISTER R1	REGISTER R2	CONSTANT c3*256+CONSTANT c4
Const	CONSTANT			CONSTAN c3
RegR	REGISTER INDIRECT		BASE REGISTER R2	DISPLACEMENT d
Reg1	1 REGISTER	REGISTER R1		
Reg0	0 OPERAND			
Cont0	PSUED 0 INSTRUCTION FOR CONTROL			
Cont1	PSUED 1 INSTRUCTION FOR CONTROL			CONSTANT c3
Cont2	PSUED 2 INSTRUCTION FOR CONTROL	CONSTANT c1	CONSTAN c2	CONSTANT c3/ArmCode ADDRESS
Cont3	PSUED 3 INSTRUCTION FOR CONTROL	ArmCode ADDRESS	ArmCode ADDRESS	ArmCode ADDRESS

FIG. 7

LOAD INSTRUCTIONS 3210

Load	LOAD ONE WORD DESIGNATED BY OPERAND 2 3 IN OPERAND 1
LoadB	LOAD ONE BYTE DESIGNATED BY OPERAND 2 3 IN OPERAND 1

MODE-WISE PROCESSING

RegReg	R1 ← [R2]
RegM1	R1 ← [R2 ^]
RegMR	R1 ← [([R2]+d) ^]
RegMD	R1 ← [d]
RegCon	R1 ← c3
RegAdr	R1 ← [R2]+d

STORE INSTRUCTIONS 3220

Store	STORE ONE WORD OF OPERAND 1 AT LOCATION DESIGNATED BY OPERAND 2, 3
StoreB	STORE ONE BYTE OF OPERAND 1 AT LOCATION DESIGNATED BY OPERAND 2 3

MODE-WISE PROCESSING

RegReg	R1 → R2
RegM1	R1 → R2 ^
RegMR	R1 → ([R2]+d) ^
RegMD	R1 → d

INTEGER-CLASS BINARY FLOATING-POINT OPERATION INSTRUCTIONS 3230

Add	ADD OPERAND 2 TO OPERAND 1
Sub	SUBTRACT OPERAND 2 FROM OPERAND
Mult	MULTIPLY OPERAND 1 BY OPERAND 2
Div	DIVIDE OPERAND 1 BY OPERAND 2
UnsAdd	SAME AS ADD EXCEPT UNSIGNED COMPUTATION
UnsSub	SAME AS SUB EXCEPT UNSIGNED COMPUTATION
AddC	SAME AS ADD EXCEPT INVOLVEMENT OF CARRY
SubB	SAME AS SUB EXCEPT INVOLVEMENT OF CARRY

MODE-WISE PROCESSING INSTRUCTIONS

RegReg	[([R1] op [R2]) → R1	op : OPERATOR
RegCon	[([R1] op c3) → R1	

FIG. 8

REAL NUMBER-CLASS BINARY FROATING-POINT OPERATION INSTRUCTIONS 3240

AddR	ADD OPERAND 2 TO OPERAND 1
SubR	SUBTRACT OPERAND 2 FROM OPERAND 1
MultR	MULTIPLY OPERAND 1 WITH OPERAND 2
DivR	DIVIDE OPERAND 1 BY OPERAND 2

MODE-WISE PROCESSING

RegReg	([R1] op [R2]) → R1	op : OPERATOR
RegCon	([R1] op c3) → R1	

BINARY LOGICAL OPERATION INSTRUCTIONS 3250

And	ANDING OF OPERAND 1 WITH OPERAND 2
Or	ORING OF OPERAND 1 WITH OPERAND 2
Xor	XORING OF OPERAND 1 WITH OPERAND 2

MODE-WISE PROCESSING

RegReg	([R1] op [R2]) → R1	op : OPERATOR
RegCon	([R1] op c3) → R1	

UNARY OPERATION INSTRUCTIONS 3260

Negate	SIGN INVERSION OF OPERAND 1
Not	BIT INVERSION OF OPERAND 1

MODE-WISE PROCESSING

Reg1	OPERATION ON CONTENT OF OPERAND 1 WITH RESULT LEFT IN OPERAND 1

COMPARISON INSTRUCTION 3270

Comp	COMPARISON OF OPERAND 1 WITH 2 TO SET CONDITION CODE

MODE-WISE PROCESSING

RegReg	IF [R1]>[R2] THEN Gt、IF [R1]=[R2] THEN Eq、IF [R1]<[R2] THEN Lt
RegCon	IF [R1]>c3 THEN Gt、IF [R1]=c3 THEN Eq、IF [R1]<c3 THEN Lt

CONDITION TEST INSTRUCTIONS 3280

TZero MODE Reg1	IF "0" THEN SET CONDITION CODE TO Eq
TBit MODE ReGCon	IF BIT INDICATED BY 3RD OPERAND OF [R1] IS "0" THEN Eq

FIG. 9

CONDITIONAL BRANCH INSTRUCTIONS 3290	
BrEq	JUMP TO POSITION DESIGNATED BY OPERAND IF CONDITION CODE IS Eq
BrNe	JUMP TO POSITION DESIGNATED BY OPERAND UNLESS CONDITION CODE IS Eq
BrGt	JUMP TO POSITION DESIGNATED BY OPERAND IF CONDITION CODE IS Gt
BrGe	JUMP TO POSITION DESIGNATED BY OPERAND IF CONDITION CODE IS Gt OR Eq
BrLe	JUMP TO POSITION DESIGNATED BY OPERAND IF CONDITION CODE IS Lt OR Eq
BrLt	JUMP TO POSITION DESIGNATED BY OPERAND IF CONDITION CODE IS Lt
IF ANY CASE, IMMEDIATELY SUCCEEDING INSTRUCTION IS EXECUTED UNLESS CONDITION IS MET	
MODE-WISE PROCESSING	
Const	POSITION DESIGNATED BY LABEL TABLE LBL [C3] IS JUMP DESTINATION INSTRUCTION ADDRESS
RegR	[R2]+d IS JUMP DESTINATION INSTRUCTION ADDRESS
UNCONDITIONAL BRANCH INSTRUCTIONS 3300	
Jump	JUMP TO POSITION DESIGNATED BY OPERAND
MODE-WISE PROCESSING	
Const	POSITION DESIGNATED BY LABEL TABLE LBL [C3] IS JUMP DESTINATION INSTRUCTION ADDRESS
RegR	[R2]+d IS JUMP DESTINATION INSTRUCTION ADDRESS
SUBPROGRAM REFERENCE INSTRUCTIONS 3310	
Call	AFTER RECORDING RETURN ADDRESS, JUMP TO POSITION DESIGNATED BY OPERAND
MODE-WISE PROCESSING	
RegMR	PLACE RETURN ASDDRESS IN R1 AND JUMP TO ADDRESS [R2]+d
RegMD	PLACE RETURN ADDRESS IN R1 AND JUMP TO ADDRESS DESIGNATED BY LBL [C3]
Reg1	AFTER STACKING RETURN ADDRESS, JUMP TO ADDRESS [R1]
RETURN INSTRUCTIONS 3320	
Return	JUMP TO RETURN ADDRESS RECORDED UPON CALL
MODE-WISE PROCESSING	
Reg1	JUMP TO RETURN ADDRESS RECORDED AT R1
Reg0	FETCH RETURN ADDRESS STACKED UPON CALL AND JUMP TO IT

FIG. 10

SHIFT INSTRUCTIONS 3330		
SHIFT CONTENT OF 1ST OPERAND R1 TO LEFT OR RIGHT BY NUMBER n INDICATED BY 2ND OR 3RD OPERAND		
ShiftL		SHIFT R1 TO LEFT BY n BITS WITH RIGHTMOST n BITS BEING "0"
ShiftR		SHIFT R1 TO RIGHT BY n BITS WITH LEFTMOST n BITS BEING "0"
MODE-WISE PROCESSING		
RegCon		R1 REPRESENTS TARGET REGISTER WITH c3 BEING BIT NUMBER FOR SHIFT
RegReg		R1 REPRESENTS TARGET REGISTER WITH [R2] BEING BIT NUMBER FOR SHIFT
ROTATION INSTRUCTIONS 3340		
ROTATE CONTENT OF 1ST OPERAND R1 TO LEFT OR RIGHT BY A NUMBER n INDICATED BY 2ND OR 3RD OPERAND		
RotL		ROTATE R1 TO LEFT BY n BITS
RotR		ROTATE R1 TO RIGHT BY n BITS
MODE-WISE PROCESSING		
RegCon		R1 REPRESENTS TARGET REGISTER WITH c3 BEING BIT NUMBER FOR THE ROTATION
RegReg		R1 REPRESENTS TARGET REGISTER WITH [R2] BEING BIT NUMBER FOR ROTATION
BIT MANIPULATION INSTRUCTIONS 3350		
GetBit	RegCC	PLACE c3-BIT DATA STARTING FROM BIT c2 OF R1 IN R1 WITH RIGHT JUSTIFICATION. LEFT PART OR RI IS "0"
GetBit	RegRCC	PLACE c3-BIT DATA STARTING FROM BIT c2 OF R2 IN R1 WITH RIGHT JUSTIFICATION. LEFT PART OR RI IS "0"
PutBit	RegRCC	PLACE RIGHTMOST c4-BIT LENGTH DATA OF R1 IN A c4-BIT LENGTH FIELD STARTING FROM BIT c3 OF R2. CONTENTS OF OTHER BIT FIELD ARE UNCHANGED
DATA CONVERTION INSTRUCTIONS 3360		
I to R	RegReg	INTEGER-TO-REAL CONVERSION OF CONTENT OF R1 AND PLACING THE RESULT IN R2
R to I	RegReg	REAL-TO-INTEGER CONVERSION OF CONTENT OF R1 AND PLACING THE RESULT IN R2
I to D	RegReg	INTEGER-TO-DOUBLE REAL CONVERSION OF CONTENT OF R1 AND PLACING THE RESULT IN R2
D to I	RegReg	DOUBLE REAL-TO-INTEGER CONVERSION OF CONTENT OF R1 AND PLACING RESULT IN R2

FIG. 11

STATE SWITCH INSTRUCTIONS 3370	
SaveSt RegMR	SAVE CURRENT PROCESSOR STATE IN MEMORY
LoadSt RegMR	RESTORE PROCESSOR STATE IN ACCORDANCE WITH INFORMATION SAVED IN MEMORY
NO-OPERATION INSTRUCTION 3380	
Nop Reg0	PERFORM NO OPERATION
PROGRAM STRUCTURE REPRESENTING PSEUDO-INSTRUCTIONS 3390	
Start Cont1 0dSym	PSEUDO-INSTRUCTION FOR STARTING OBJECT HAVING NAME (OF SYMBOL LABEL) INDICATED BY c3
SubP Cont1 OdSym	START OF SUBPROGRAM
Block Cont0	START BLOCK
End Cont0	END BLOCK
Loops Cont1 0dLab	INDICATE HEAD OF LOOP STATEMENT HAVING REPETITION STARTING POINT AT LABEL INDICATED BY c3
Loope Cont1 0dLab	INDICATE TRAIL OF LOOP STATEMENT HAVING REPETITION STARTING POINT AT LABEL INDICATED BY c3
Pend Cont0	END SUBPROGRAM OR UNIT
Stmt Cont1 0dCi	INDICATE START POSITION OF STATEMENT HAVING c3 AS STATEMENT NUMBER
SYMBOL NAME DESIGNATING PSEUDO-INSTRUCTIONS 3400	
Eentry Cont1 0dSym	PSEUDO-INSTRUCTION INDICATING NAME (OF SYMBOL TABLE) INDICATED BY c3 AS ENTRY NAME
Extern Cont1 0dSym	PSEUDO-INSTRUCTION INDICATING THAT NAME (OF SYMBOL TABLE) INDICATED BY c3 IS EXTERNAL NAME
Label Contl 0dLab	DEFINE LABEL INDICATED BY LABEL TABLE c3
Label Cont1 0dCL	DEFINE NAME OF MEMORY CONSTANT INDICATED BY LOGICAL REGISTER NUMBER c3
Label Cont1 0dGvar	DEFINE GENERATED VARIABLE NAME INDICATED BY c3 AS LABEL NAME
Name Cont1 0dSym	INSTRUCTION FOR EQUATING NAME INDICATED BY c3 TO THE NAME OF IMMEDIATELY PRECEDING LABEL

FIG. 12

MEMORY DESIGNATING PSEUDO-INSTRUCTIONS 3410	
Dconst Cont1 0dCi	CONSTANT DEFINITION DEFINING c3 AS INTEGER CONSTANT VALUE
Dconst Cont1 0dCS	CONSTANT DEFINITION REGARDING c3 AS ONE-WORD CHARACTER STRING
Dconst Cont1 0dCS	CONSTANT DEFINITION REGARDING c3 AS ONE-WORD CHARACTER STRING CONSTANT
Dword Cont1 0dCI	DEFINE STORAGE INSTRUCTION FOR SECURING MEMORY OF c3 WORDS
Daddr Cont1 0dSym	DEFINE ADDRESS FOR THE NAME (OF SYMBOL TABLE) INDICATED BY c3
MCode Cont1 0dcI	INSTRUCTION FOR GENERATING MACHINE LANGUAGE MRT[c3] INDICATED BY c3
PSEUDO-INSTRUCTIONS DESIGNATING DEBUGGER-ORIENTED SYMBOL INFORMATION, ETC. 3420	
PInf Cont1 0dCi	INDICATE INFORMATION OF PROGRAM CHARACTERISTIC INFORMATION TABLE RESIDENT AT POSITION INDICATED BY c3
SInf Cont1 0dCi	INDICATE SYMBOL INFORMATION RESIDENT AT POSITION INDICATED BY c3

FIG. 13

```
 1   int x ;                                  3610
 2   int a [10], b [10], c [10] ;
 3   func ()                                  3614
 4     { int i, n ;                           3616
         ...
                                              3620
10       for (i=0 ; i<n ; ++i) {
11       if  (a [i]>0  {
12         if  (a [i]<x)
13           b [i]=1;
           }
14       else c [i]=1 ;
         ...
         }
```

FIG. 14

```
    SInf (global, var,                                          ~3700
          ((x, 1, int, 4),
           (a, 2, int, 40, array, 10),
           (b, 2, int, 40, array, 10),
           (c, 2, int, 40, array, 10))) ;
    SInf (local, func, var,                                     ~3702
          ((i, 4, int, 4), (n, 4, int, 4))) ;
    ...
        alloc (Vbase,RcSect,AlcVb,0,0,0x62489700                ~ 3710
                                    +0x00224897) ;
        bblock (1, pred ( ), succ (2)) ;                        ~ 3712
    Stmt   Cont1  Noreg Noreg 0dCi 10 ;
    Loops  Cont1  Noreg Noreg 0dLab L1 ;                        ~3716
        alloc (Ar5,RcArith,Alc5,0,0,0xD1044000                  ~3718
                                    +0x00110440) ;
*   Load   RegCon Ar5  Noreg 0dCi 0 ;                           ~3720
*   Store  RegMr  Ar5  Vbase 0dDisp i ;                         ~3722
        bblock (2,  pred (1,8), succ (3,7)) ;                   ~3724
    Block Cont0  Noreg  Noreg 0dNo Null ;
    Lable Cont1  Noreg  Noreg 0dLab L1 ;                        ~3728
    Stmt  Cont1  Noreg  Noreg 0dCi 11 ;
        alloc (Ar6,RcAddr,Alc6,2,0,0xE0000000) ;                ~3732
*   Load   RegReg Ar6  Vbase 0dNo Null ;                        ~3734
*   Add    RegReg Ar6  Ar5   0dNo Null ;                        ~3736
        alloc (Ar7,RcArith,Alc7,4,0,0xC0000000) ;
*   Load   RegMr  Ar7  Ar6   0dDisp  a ;                        ~3740
        free (Ar6) ;
*   Comp   RegCon Ar7  Noreg 0dCi 0 ;
        free (Ar7) ;
    BrLe   Const  Noreg Noreg 0dLab L2 ;                        ~3748
```

FIG. 15

```
        bblock (3, pred (2), succ (4,5)) ;                          ~3750
    Block  Cont0  Noreg Noreg  0dNo  Null ;
    Stmt   Cont1  Noreg Noreg  0dCi  12 :
        alloc (Ar8, RcAddr,Alc8,6,0,0xE0000000) ;
*   Load  RegReg Ar8   Vbase  0dNo  Null ;                          ~3758
*   Add   RegReg Ar8   Ar5 0dNo  Null ; /* Same as Ar6 */          ~3760
        alloc (Ar9, RcArith,Alc9,8,0,0xA0000000) ;
*   Load  RegMr  Ar9   Ar8 0dDisp  a ; /* Same as Ar7 */           ~3770
        free (Ar8) ;
        alloc (Ar10, RcArith,Alc10,9,0,0xC0000000) ;
*   Load  RegMr  Ar10  Vbase  0dDisp  x ;
*   Comp RegReg Ar9    Ar10   0dNo  Null ;
        free (Ar9) ;
        free (Ar10) ;
    BrGe  Const  Noreg Noreg 0dLab L3 ;                             ~3778
        bblock (4,  pred (3),  succ (5,6)) ;                        ~3780
    Stmt  Cont1   Noreg Noreg  0dCi  13 ;
        alloc (Ar11,RcArith,Alc11,11,0,0x90000000) ;
*   Load  RegCon  Ar11  Noreg  0dCi 1 ;                             ~3786
        alloc (Ar12,RcAddr,Alc12,12,0,0xC0000000) ;
*   Load  RegReg  Ar12  Vbase  0dNo Null ;                          ~3790
*   Add   RegReg  Ar12  Ar5  0dNo Null ; /* Same as Ar6 */          ~3792
*   Store RegMr   Ar11  Ar12 0dDisp b ;
        free (Ar12) ;
        free (Ar11) ;
    End  Cont0  Noreg  Noreg  0dNo  Null ;
```

FIG. 16

```
    Jump  Const  Noreg  Noreg  0dLab  L4 ;                    ~3802
        bblock (5, pred (3), succ (8)) ;
    Label  Cont1  Noreg  Noreg  0dLab  L3 ;                   ~3806
    Jump  Const  Noreg  Noreg  0dLab  L5 ;                    ~3808
        bblock (6,  pred (4,  succ (8)) ;
    Label  Cont1  Noreg  Noreg  0dLab  L4 ;                   ~3812
    Jump  Const  Noreg  Noreg  0dLab  L5 ;                    ~3814
        bblock (7,  pred (2),  succ (8)) ;
    Label  Cont1  Noreg  Noreg  0dLab  L2 ;
    Stmt  Cont1  Noreg  Noreg  0dCi  14 ;
        alloc (Ar13, RcArith, Alc13,15,0,0x90000000) ;
*   Load RegCon Ar13  Noreg  0dCi 1 ; /* Same as Ar11 */
        alloc (Ar14, RcAddr, Alx14,16,0,0xE0000000) ;         ~3828
*   Load RegReg Ar14  Vbase  0dNo Null ;
*   Add  RegReg Ar14  Ar5   0dNo Null ; /* Same as Ar6 */    ~2830
*   Store RegMr  AR13 Ar14  0dDISP c ;
        free (Ar13) ;
        free (Ar14) ;
    End   Cont0  Noreg  Noreg  0dNo  Null ;
        bblock (8, pred (5,6,7), succ (2,9)) ;               ~3840
    Label  Cont1  Noreg  Noreg  0dLab  L5 ;
        alloc (Ar15, RcArith, Alc15,19,0,0xE0000000) ;
*   Load  RegMr  Ar15  Vbase  0dDisp j ; /* Same as Ar5*/     ~3846
*   Add  RegCon  Ar15  Noreg  0dCi 1 ;
*   Store  RegMr  Ar15  Vbase  0dDisp  i ;                   ~3850
        free (Ar15) ;
        alloc (Ar16,RcArith,Alc16,22,0,0xA0000000) ;
*   Load  RegMr  Ar16  Vbase  0dDidp i ; /* Same as Ar15 */  ~3856
        alloc (Ar17,RcArith,Alc17,23,0,0xC0000000) ;
*   Load  RegMr  Ar17  Vbase  0dDisp n ;                     ~3860
*   Comp RegReg Ar16  Ar17   0dNo Null ;
        free (Ar16) ;
        free (Ar17) ;
    BrLe  Const  Noreg  Noreg  0dLab  L1 ;                   ~3868
        free (Ar5) ;
        bblock (9, pred (8), succ (10)) ;
    Loope Cont1  Noreg  Noreg  0dLab  L1 ;                   ~3874
    ...
```

FIG. 17

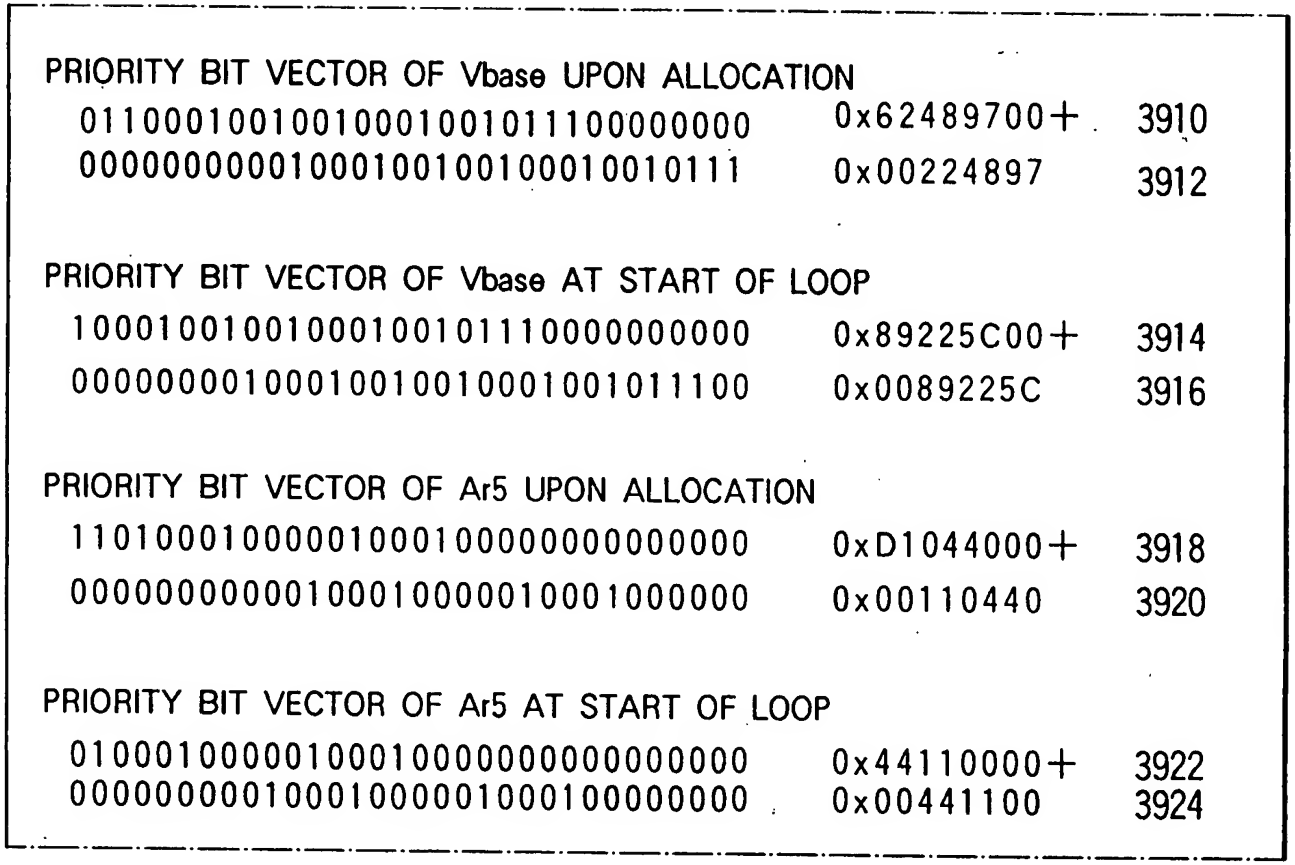

PRIORITY BIT VECTOR OF Vbase UPON ALLOCATION
01100010010010001001011100000000
0x62489700+
3910
00000000001000100100100010010111
0x00224897
3912
PRIORITY BIT VECTOR OF Vbase AT START OF LOOP
10001001001000100101110000000000
0x89225C00+
3914
00000000100010010010001001011100
0x0089225C
3916
PRIORITY BIT VECTOR OF Ar5 UPON ALLOCATION
11010001000001000100000000000000
0xD1044000+
3918
00000000000100010000010001000000
0x00110440
3920
PRIORITY BIT VECTOR OF Ar5 AT START OF LOOP
01000100000100010000000000000000
0x44110000+
3922
00000000010001000001000100000000
0x00441100
3924

FIG. 18

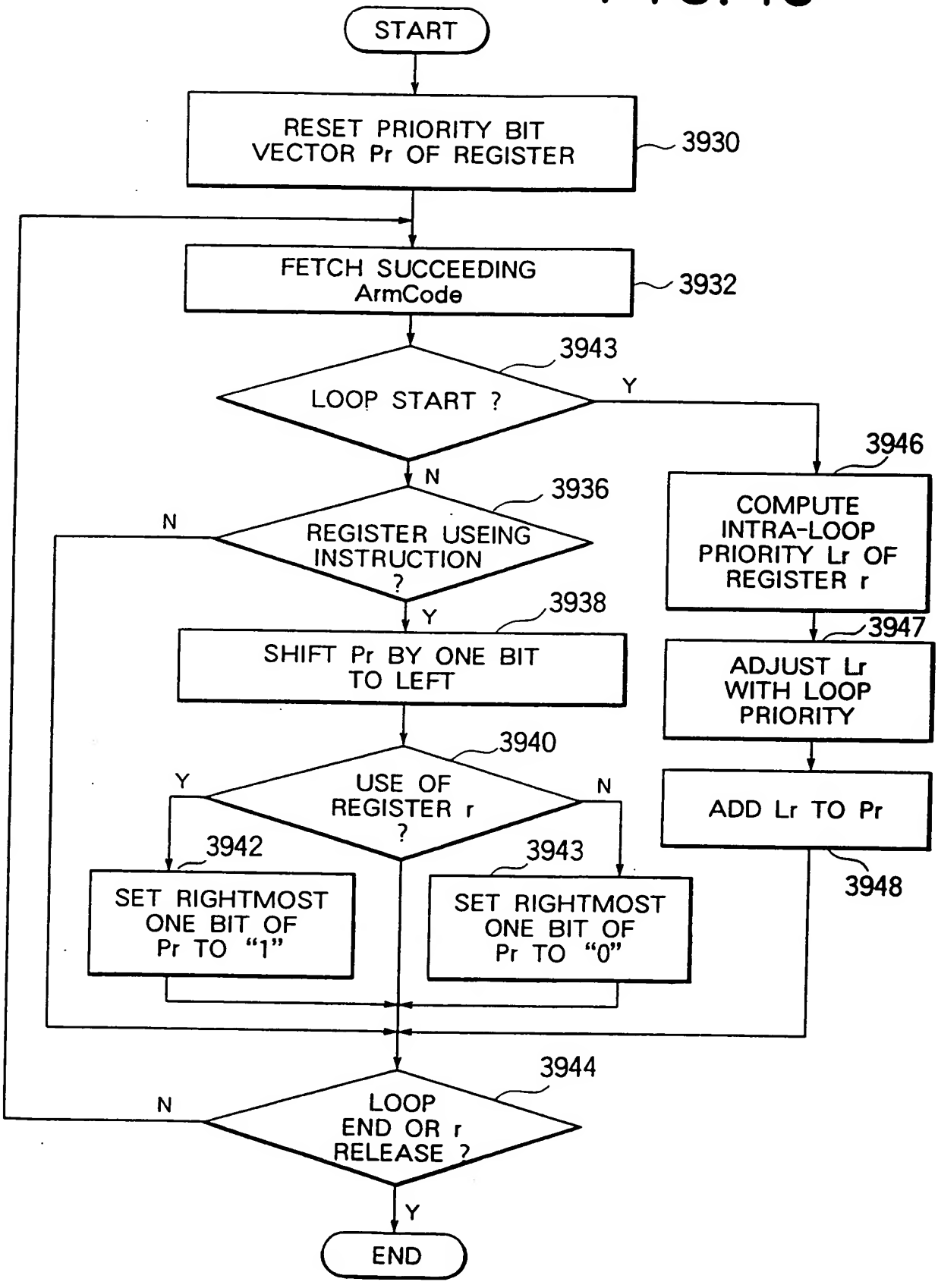

START
RESET PRIORITY BIT VECTOR Pr OF REGISTER
3930
FETCH SUCCEEDING ArmCode
3932
3943
LOOP START ?
Y
N
3946
COMPUTE INTRA-LOOP PRIORITY Lr OF REGISTER r
3936
REGISTER USEING INSTRUCTION ?
N
Y
3947
ADJUST Lr WITH LOOP PRIORITY
3938
SHIFT Pr BY ONE BIT TO LEFT
3940
USE OF REGISTER r ?
Y
N
ADD Lr TO Pr
3948
3942
SET RIGHTMOST ONE BIT OF Pr TO "1"
3943
SET RIGHTMOST ONE BIT OF Pr TO "0"
3944
LOOP END OR r RELEASE ?
N
Y
END

FIG. 19

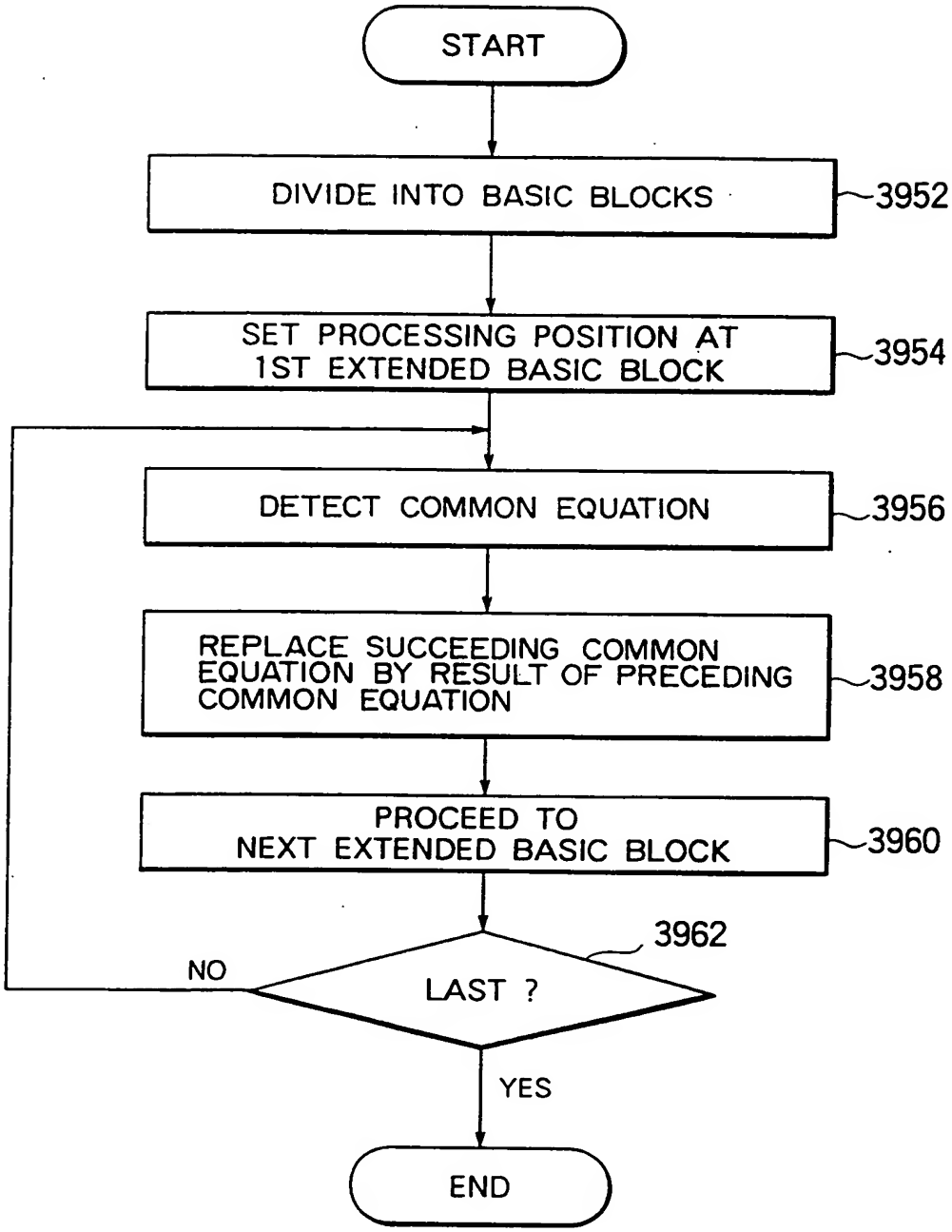

START
DIVIDE INTO BASIC BLOCKS
3952
SET PROCESSING POSITION AT 1ST EXTENDED BASIC BLOCK
3954
DETECT COMMON EQUATION
3956
REPLACE SUCCEEDING COMMON EQUATION BY RESULT OF PRECEDING COMMON EQUATION
3958
PROCEED TO NEXT EXTENDED BASIC BLOCK
3960
3962
LAST ?
NO
YES
END

FIG. 20

```
    SInf (global, var,
          ((x, 1, int, 4),
           (a, 2, int, 40, array, 10),
           (b, 2, int, 40, array, 10),
           (c, 2, int, 40, array, 10))) ;
    SInf (local, func, var,
          ((i, 4, int, 4), (n, 4, int, 4))) ;
    ...
        alloc (Vbase,RcSect,AlcVb,0,0,0x5C040000
                                    +0x00404000) ;
        bblock (1, pred ( ), succ (2)) ;          Vbase
    Stmt   Cont1  Noreg Noreg 0dCi 10 ;            |
    Loops  Cont1  Noreg Noreg 0dLab L1 ;           |
        alloc (Ar5,RcArith,Alc5,0,0,0xC20E0000+0x0020E000) ;     ~4028
*   Load   RegCon Ar5  Noreg 0dCi 0 ;              |  5          ~4030
*   Store  RegMr  Ar5  Vbase 0dDisp i ;           Vb  |          ~4032
        alloc (Ar11,RcArith,Alc11,2,4,0x80800000+0x00080000) ;
        if  (Alc11)  {                             |  |
*   Load   RegCon Ar11  Noreg 0cCi 1 ;}            |  |  11      ~4038
        alloc (Ar10,RcArith,Alc10,3,3,0x82000000+0x00200000) ;
        if  (Alc10)  {                             |  |  |
*   Load   RegMr  Ar10 Vbase 0dDisp x ;}          Vb  |  |  10  ~4044
        alloc (Ar17,RcArith,Alc17,4,2,0x80200000+0x00020000) ;   ~4046
        if  (Alc17)  {                             |  |  |  |   ~4048
*   Load   RegMr  Ar17 Vbase 0dDisp n ;}          Vb  |  |  | 17
                                                                 ~4050
```

FIG. 21

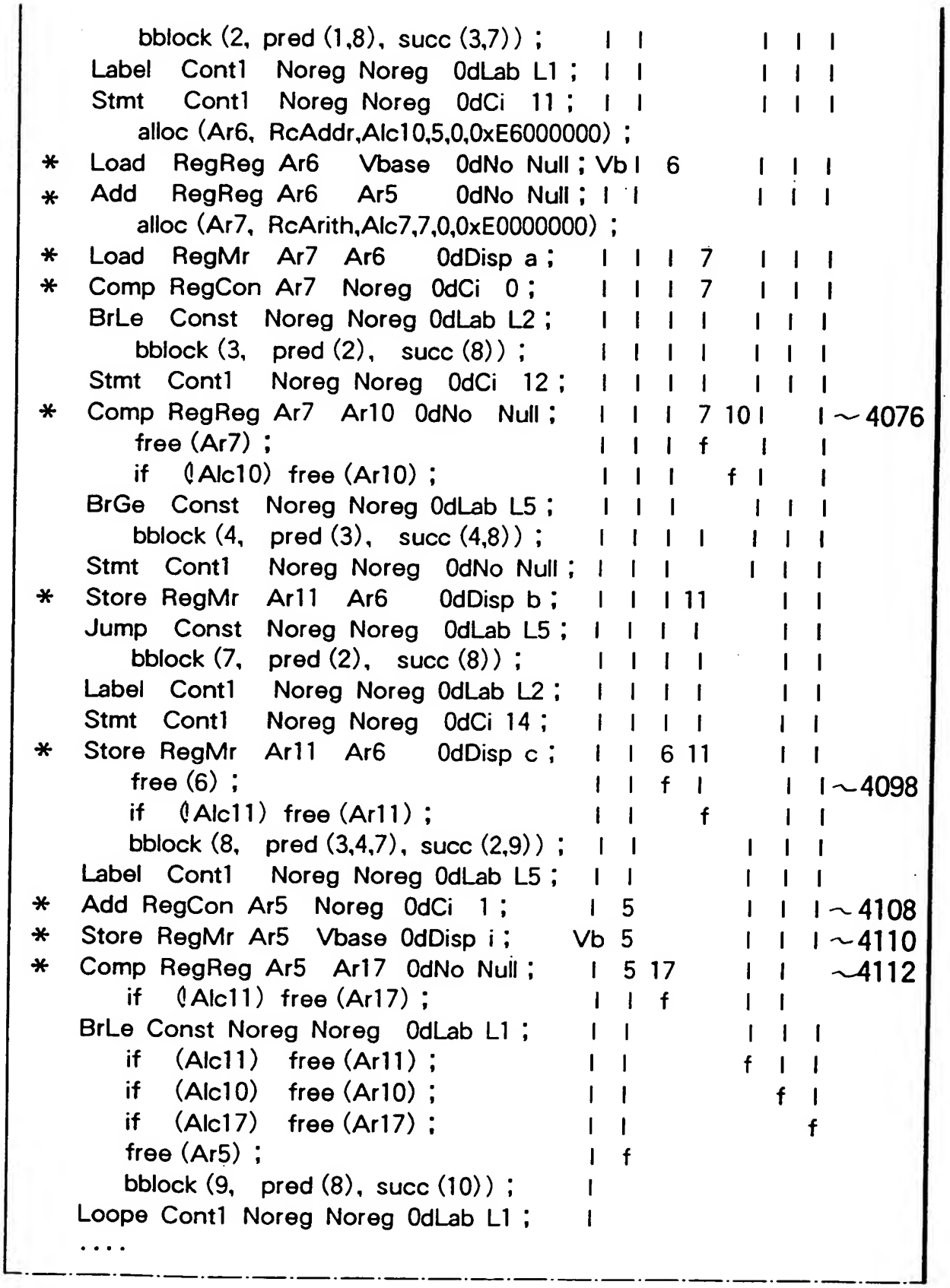

bblock (2, pred (1,8), succ (3,7)) ;
Label Cont1 Noreg Noreg 0dLab L1 ;
Stmt Cont1 Noreg Noreg 0dCi 11 ;
alloc (Ar6, RcAddr,Alc10,5,0,0xE6000000) ;
* Load RegReg Ar6 Vbase 0dNo Null ;
* Add RegReg Ar6 Ar5 0dNo Null ;
alloc (Ar7, RcArith,Alc7,7,0,0xE0000000) ;
* Load RegMr Ar7 Ar6 0dDisp a ;
* Comp RegCon Ar7 Noreg 0dCi 0 ;
BrLe Const Noreg Noreg 0dLab L2 ;
bblock (3, pred (2), succ (8)) ;
Stmt Cont1 Noreg Noreg 0dCi 12 ;
* Comp RegReg Ar7 Ar10 0dNo Null ;
free (Ar7) ;
if (!Alc10) free (Ar10) ;
BrGe Const Noreg Noreg 0dLab L5 ;
bblock (4, pred (3), succ (4,8)) ;
Stmt Cont1 Noreg Noreg 0dNo Null ;
* Store RegMr Ar11 Ar6 0dDisp b ;
Jump Const Noreg Noreg 0dLab L5 ;
bblock (7, pred (2), succ (8)) ;
Label Cont1 Noreg Noreg 0dLab L2 ;
Stmt Cont1 Noreg Noreg 0dCi 14 ;
* Store RegMr Ar11 Ar6 0dDisp c ;
free (6) ;
if (!Alc11) free (Ar11) ;
bblock (8, pred (3,4,7), succ (2,9)) ;
Label Cont1 Noreg Noreg 0dLab L5 ;
* Add RegCon Ar5 Noreg 0dCi 1 ;
* Store RegMr Ar5 Vbase 0dDisp i ;
* Comp RegReg Ar5 Ar17 0dNo Null ;
if (!Alc11) free (Ar17) ;
BrLe Const Noreg Noreg 0dLab L1 ;
if (Alc11) free (Ar11) ;
if (Alc10) free (Ar10) ;
if (Alc17) free (Ar17) ;
free (Ar5) ;
bblock (9, pred (8), succ (10)) ;
Loope Cont1 Noreg Noreg 0dLab L1 ;
....
~4076
~4098
~4108
~4110
~4112

FIG. 22

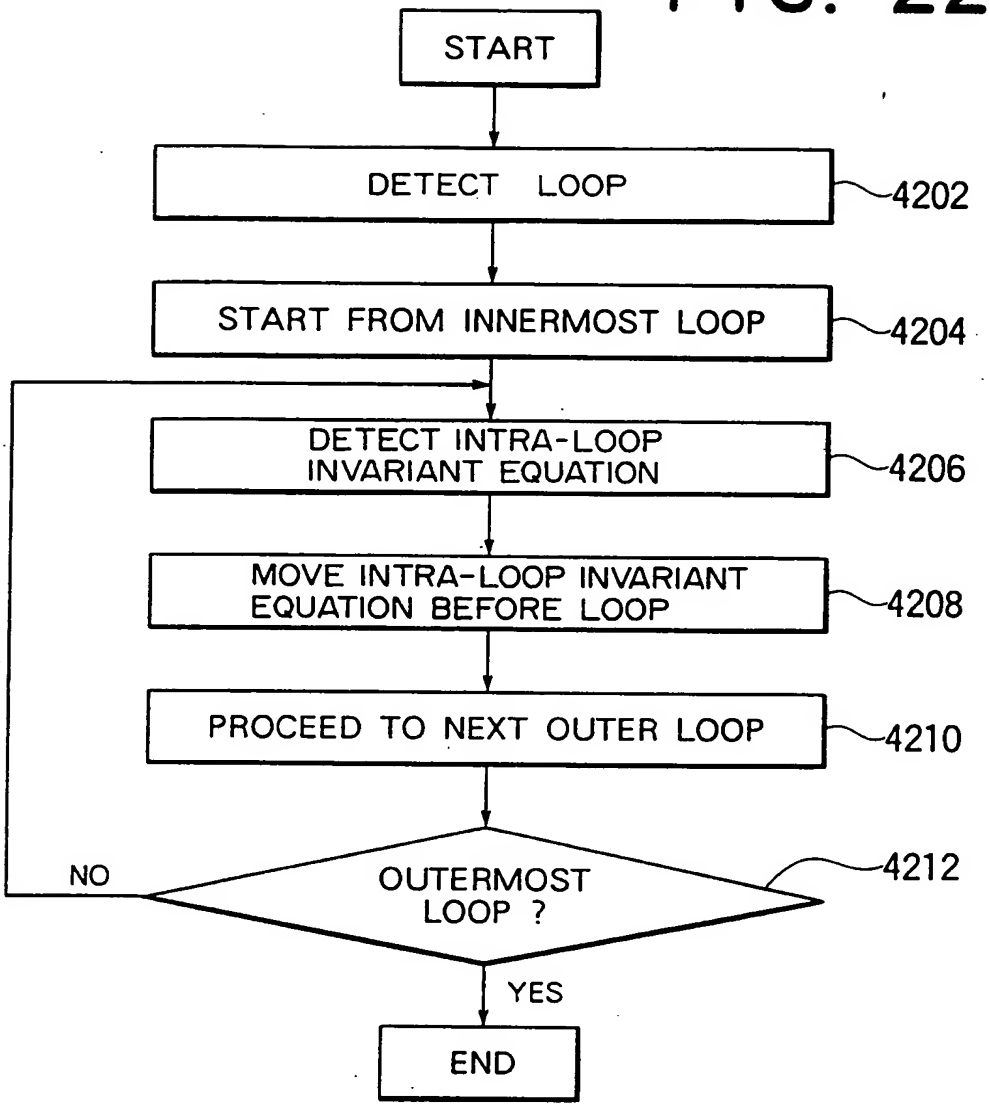


FIG. 23

```
RcArith   =[ 8 . . 23] ;                          ~4260
RcAddr    =[16 . . 23] ;                          ~4262
RcSect    =[20 . . 23] ;                          ~4264
RcReturn  =[31] ;                                 ~4266
RcFuncval =[ 2 . .  3] ;                          ~4268
RcParm    =[ 4 . .  7] ;                          ~4270
RcNosave  =[ 8 . . 15, 24, 25] ;                  ~4272
RcTemp    =[24 . . 25] ;                          ~4274
RcFixed   =[ 0, 1, 26, 27, 28, 29, 30, 31] ;      ~4276
RcAny     =[ 0 . . 31] ;                          ~4278
```

FIG. 24

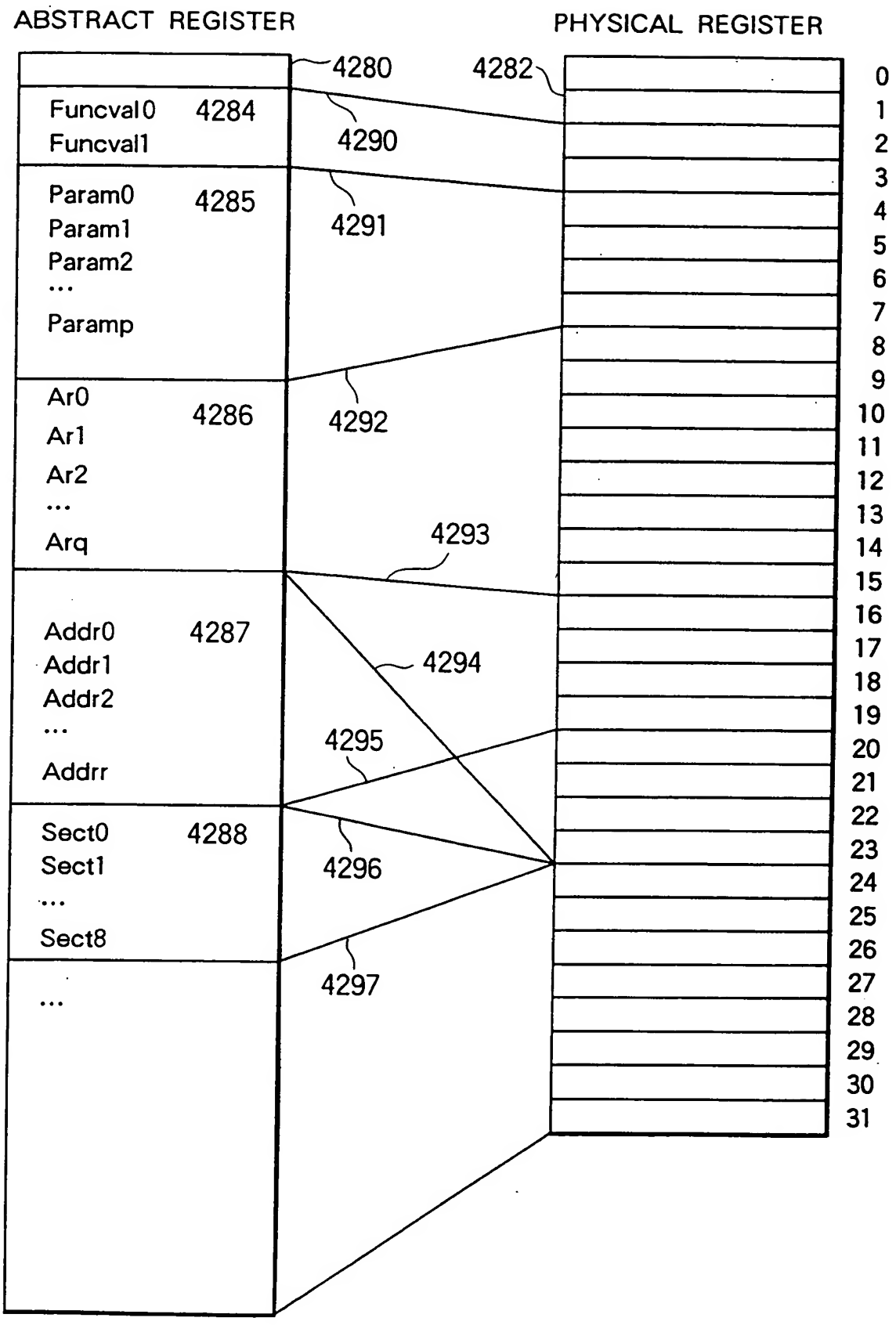

ABSTRACT REGISTER
PHYSICAL REGISTER
4280
4282
Funcval0
Funcval1
4284
4290
Param0
Param1
Param2
...
Paramp
4285
4291
Ar0
Ar1
Ar2
...
Arq
4286
4292
4293
Addr0
Addr1
Addr2
...
Addrr
4287
4294
4295
Sect0
Sect1
...
Sect8
4288
4296
4297
...
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

FIG. 25

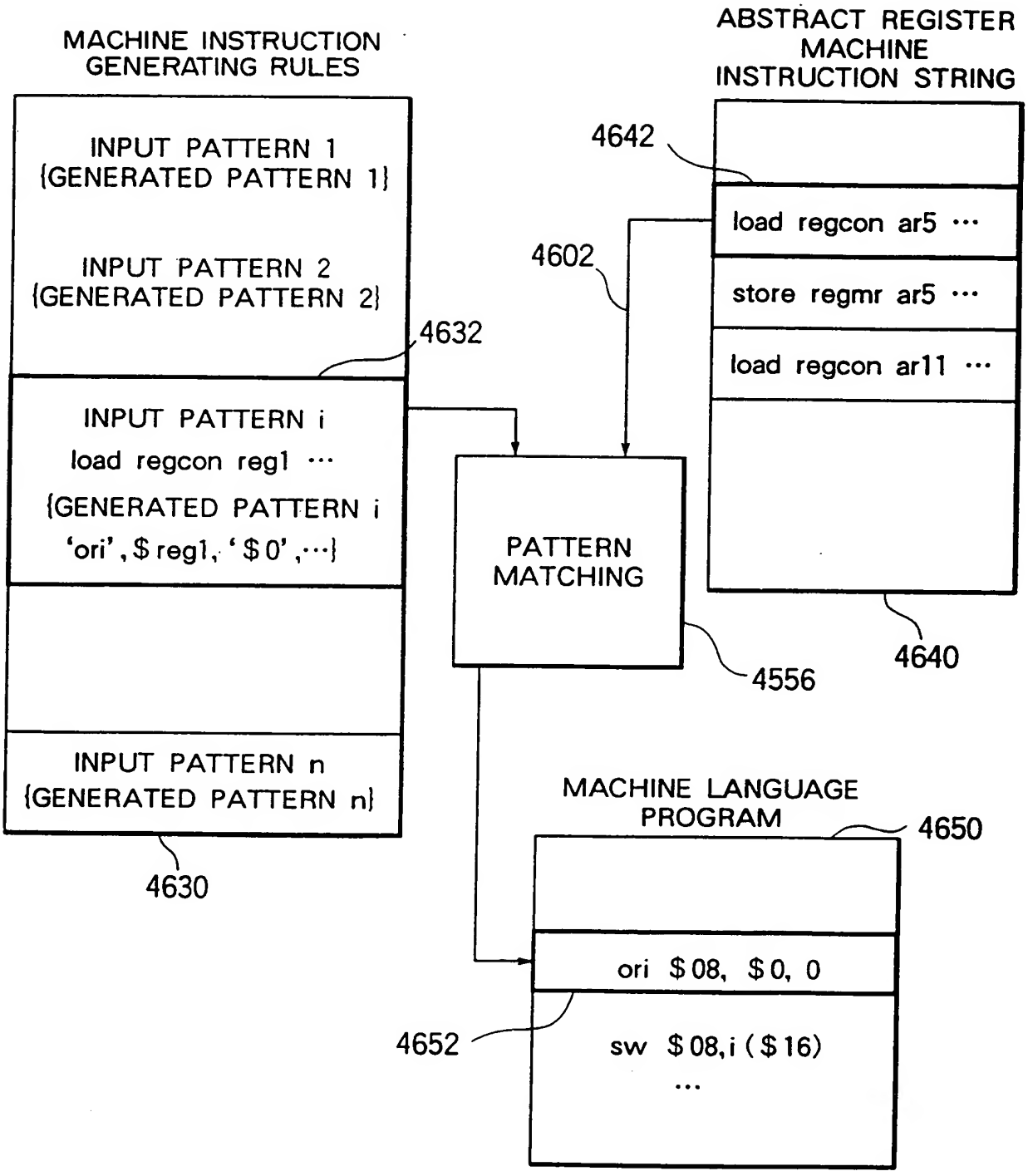

MACHINE INSTRUCTION GENERATING RULES
INPUT PATTERN 1
{GENERATED PATTERN 1}
INPUT PATTERN 2
{GENERATED PATTERN 2}
INPUT PATTERN i
load regcon reg1 ···
{GENERATED PATTERN i
'ori', $ reg1, '$ 0', ···}
INPUT PATTERN n
{GENERATED PATTERN n}
4630
4632
ABSTRACT REGISTER MACHINE INSTRUCTION STRING
4642
load regcon ar5 ···
store regmr ar5 ···
load regcon ar11 ···
4640
4602
PATTERN MATCHING
4556
MACHINE LANGUAGE PROGRAM
4650
ori $08, $0, 0
sw $08,i ($16)
···
4652

FIG. 26

```
    Load RegCon Reg1 Noreg 0dCi C                                    ~4300
      {if ($C<65536) && ($C >= 0)                                    ~4302
            gen (' ori ', $Reg1, ' $0', $C, null, null) ;            ~4304
        else {
            gen (' iui', $Reg1, $C/65536, null, null, null) ;        ~4308
            gen (' ori ',$Reg1, $Reg1, $C %65536, null, null) ; } ;  ~4310
      }
|   Load RegMr Reg1 Reg2 0dDisp V                                    ~4312
      {gen (' lw', $Reg1, $V, ' (',$Reg2, ')') ;
      }
|   Load RegReg Reg1 Reg2 0dNo Null                                  ~4316
    Add  RegReg Reg1 Reg3 0dNo Null
      {gen (' add ', $Reg3, $Reg2, $Reg1, null, null) ;
      }
|   Store RegMr Reg1 Reg2 0dDisp V
      {gen (' sw ', $Reg1, $V, ' (', $Reg2, ')') ;
      }
|   Add RegCon Reg1 NoReg 0dCi C
      {if ($C<32768) && ($C >= -32768)
        gen (' addi ', $Reg1, $Reg1, $C, null, null) ;
       else {
        gen (' lw ', ' $24', '=', $C, null, null) ;
        gen (' add', $Reg1, $Reg1, ' $24', null, null) ; } ;
      }
|   Comp RegCon Reg1 Noreg 0dCi 0
    BrLe Const Noreg Noreg 0dLab L
      {gen (' blez ', $Reg1, Label ($L), null, null, null) ;
      }
|   Comp RegReg Real Reg3 0dNo Null
    BrGe Const Noreg Noreg 0dLab L
      {gen (' sub ', $Reg1, $Reg2, ' $24', null, null) ;
       gen (' bgez',' $24', Label ($L), null, null, null) ;
      }
|   Jump Const Noreg Noreg 0dLab L
      {gen (' j ', Label (&L), null, null, null, null) ;
      }
```

FIG. 27

```
Ar5    -> $08, Ar6  -> $17, Ar7  -> $09, ~4400
Ar10   -> $10, Ar11 -> $11, Ar17 -> $12,
Vbase  -> $16
```

FIG. 28

```
        ori     $08,$0, 0        ~4500
        sw      $08, i($16)
        ori     $11,$0, 1
        lw      $10, x($16)
        lw      $12, n($16)
L01 :   add     $16,$08,$17
        lw      $09, a($17)
        blez    $09, L02
        sub     $24,$9,$10
        bgez    $24, L05
        sw      $11, b($17)
        j       L05
L02 :   sw      $11, c($17)
L05 :   addi    $08,$08, 1
        sub     $24,$08,$12      ~4526
        blez    $24, L01
```

FIG. 29

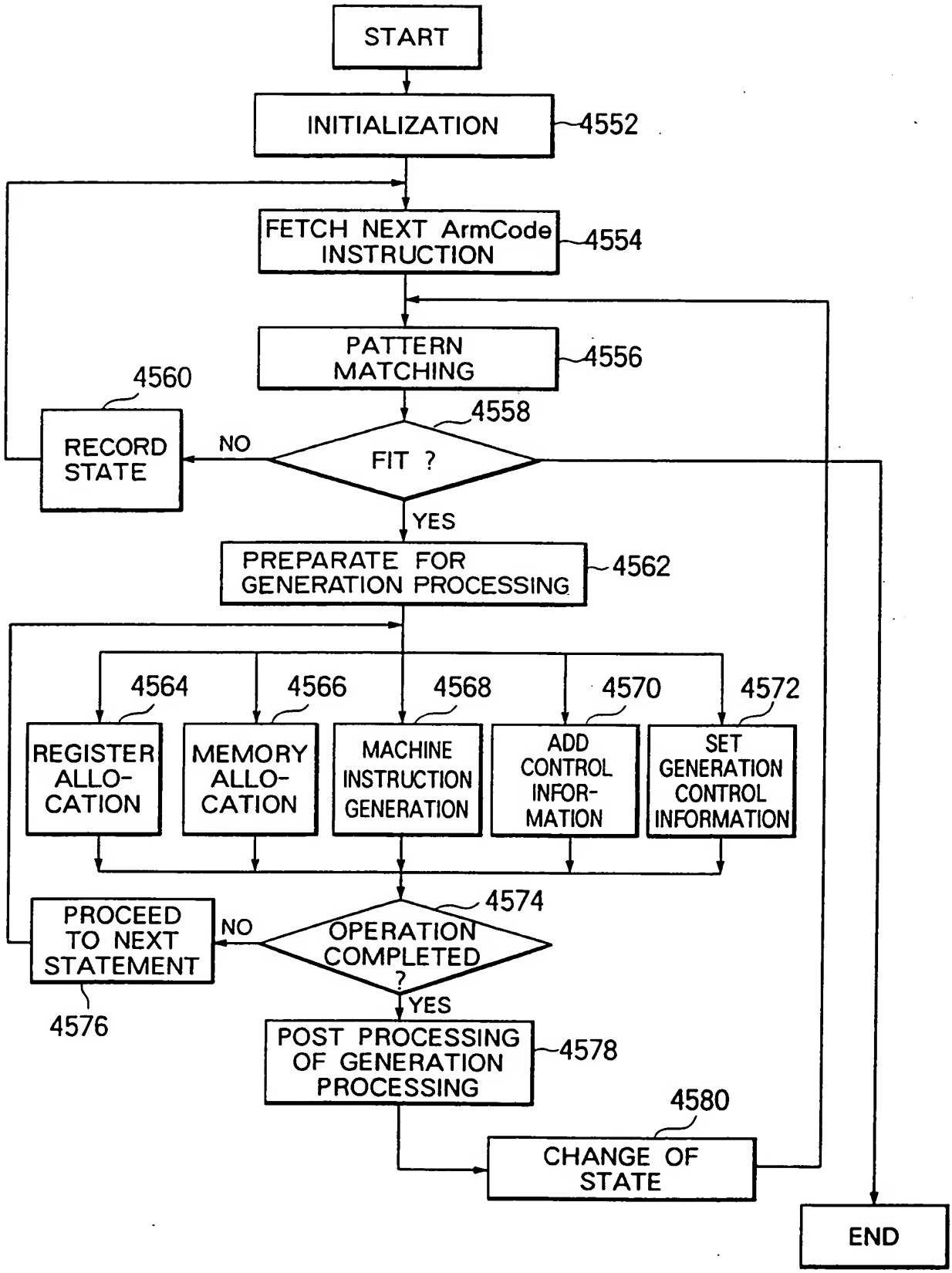

START
INITIALIZATION
4552
FETCH NEXT ArmCode INSTRUCTION
4554
PATTERN MATCHING
4556
4560
RECORD STATE
NO
4558
FIT ?
YES
PREPARATE FOR GENERATION PROCESSING
4562
4564
4566
4568
4570
4572
REGISTER ALLO-CATION
MEMORY ALLO-CATION
MACHINE INSTRUCTION GENERATION
ADD CONTROL INFOR-MATION
SET GENERATION CONTROL INFORMATION
PROCEED TO NEXT STATEMENT
NO
4574
OPERATION COMPLETED ?
YES
4576
POST PROCESSING OF GENERATION PROCESSING
4578
4580
CHANGE OF STATE
END

FIG. 30

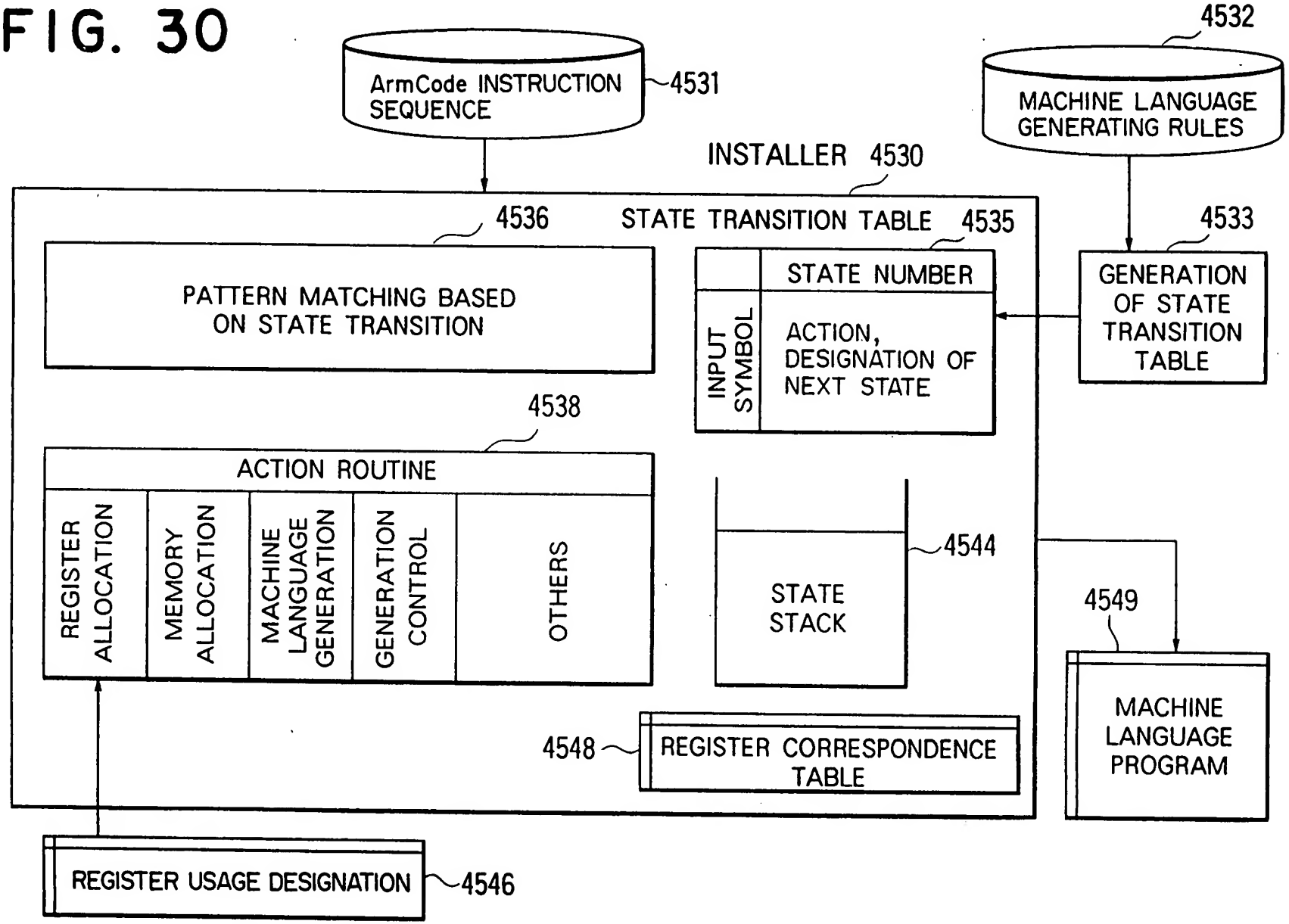

ArmCode INSTRUCTION SEQUENCE
4531
4532
MACHINE LANGUAGE GENERATING RULES
INSTALLER 4530
4536
STATE TRANSITION TABLE 4535
4533
PATTERN MATCHING BASED ON STATE TRANSITION
STATE NUMBER
INPUT SYMBOL
ACTION, DESIGNATION OF NEXT STATE
GENERATION OF STATE TRANSITION TABLE
4538
ACTION ROUTINE
REGISTER ALLOCATION
MEMORY ALLOCATION
MACHINE LANGUAGE GENERATION
GENERATION CONTROL
OTHERS
STATE STACK
4544
4549
MACHINE LANGUAGE PROGRAM
4548
REGISTER CORRESPONDENCE TABLE
REGISTER USAGE DESIGNATION
4546

FIG. 31

```
RcArith     =[1. . 7, 16. . 23] ;      ~4600
RcAddr      =[1. . 7, 16. . 23] ;      ~4602
RcSect      =[1. . 7] ;                ~4604
RcReturn    =[31] ;
RcFuncval   =[8. . 13] ;
RcParm      =[24. . 29] ;
RcNosave    =[16. . 23] ;
RcTemp      =[15] ;                    ~4616
RcFixed     =[0, 14, 30, 31] ;
RcAny       =[0. . 31] ;
```

FIG. 32

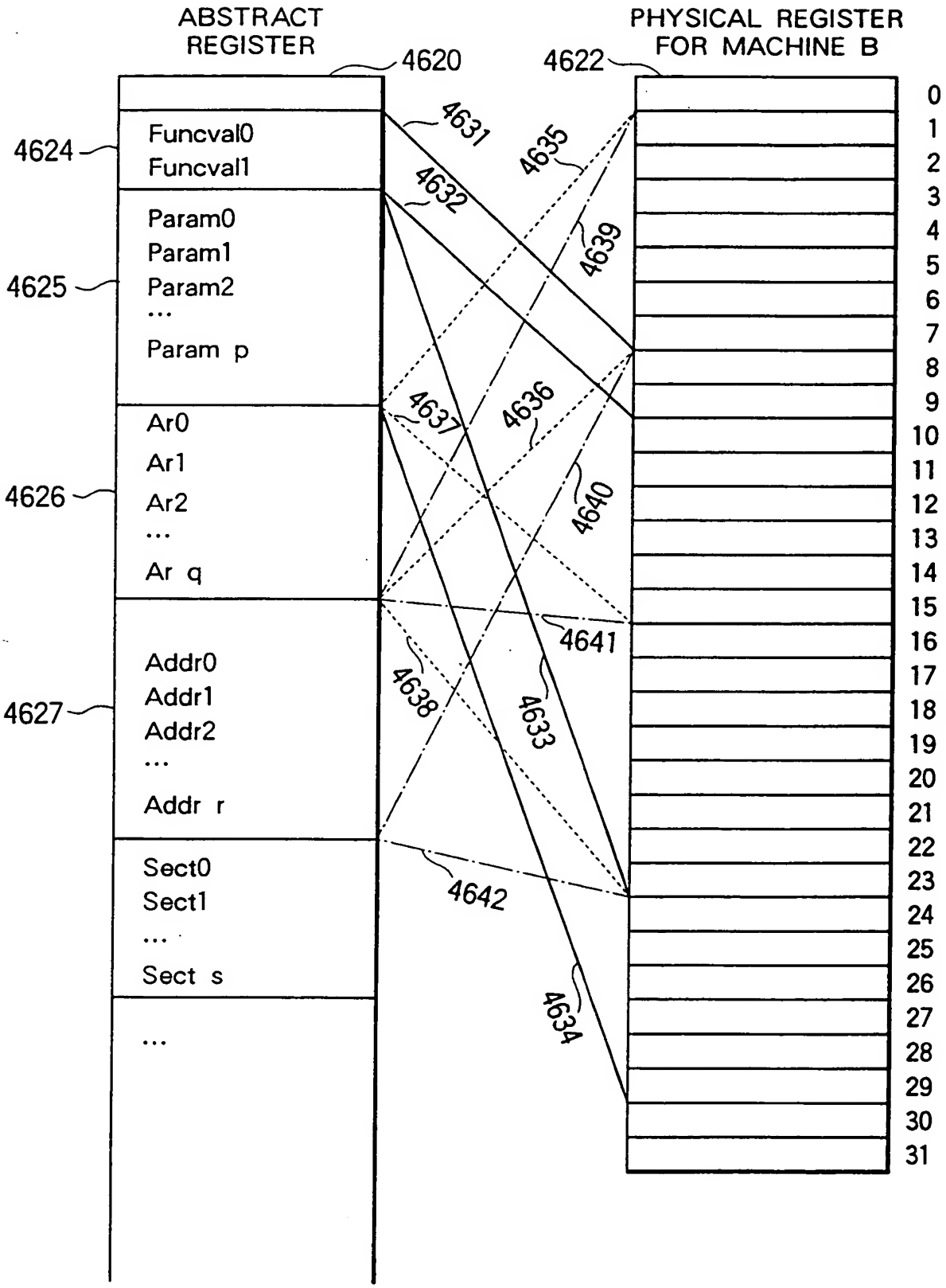

ABSTRACT REGISTER
4620
PHYSICAL REGISTER FOR MACHINE B
4622
4624
Funcval0
Funcval1
4625
Param0
Param1
Param2
...
Param p
4626
Ar0
Ar1
Ar2
...
Ar q
4627
Addr0
Addr1
Addr2
...
Addr r
Sect0
Sect1
...
Sect s
...
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

FIG. 33

```
Load RegCon Reg1 Noreg 0dCi C                                    ~4700
  {if ($C<4096) && ($C >= -4096)
       gen ('or', '%0', $Reg1, null, null) ;
    else {
       gen ('sethi ', $C/4096, $Reg1, null, null, null) ;
       gen ('or ',$Reg1, $C % 4096, $Reg1, null, null) ; } ;
  }
| Load RegMr Reg1 Reg2 0dDisp V                                   ~4712
  {gen ('ld', $Reg2, '+', $V, $Reg1, null) ;
  }
| Load RegReg Reg1 Reg2 0dNo Null                                 ~4716
  Add RegReg Reg1 Reg3 0dNo Null
  {gen ('add ', $Reg2, $Reg3, $Reg1, null, null) ;
  }
| Store RegMr Reg1 Reg2 0dDisp V                                  ~4722
  {gen ('st ', $Reg1, $Reg2, '+', $V, null) ;
  }
| Add RegCon Reg1 NoReg 0dCi C
  {if ($C<4096) && ($C >= -4096)
    gen ('add ', '%0', $C, $Reg1, null, null) ;
   else {
    gen ('id ', '=', $C, '%15', null, null) ;
    gen ('add', $Reg1, '%15', $Reg1, null, null) ; } ;
  }
| Comp RegCon Reg1 Noreg 0dCi 0
  BrLe Const Noreg Noreg 0dLab L
  {gen ('sub ', $Reg1, '%0', $Reg1, null, null) ;
   gen ('ble  ', Label ($L), null, null, null, null) ;
  }
| Comp RegReg Real Reg3 0dNo Null
  BrGe Const Noreg Noreg 0dLab L
  {gen ('sub ', $Reg1, $Reg2, ''%15', null, null) ;
   gen ('bge  ', Label ($L), null, null, null, null) ;
  }
| Jump Const Noreg Noreg 0dLab L
  {gen ('ba ', Label (&L), null, null, null, null) ;
  }
```

FIG. 34

```
Ar5  -> %16, Ar6  -> %20, Ar7  -> %21, ~4780
Ar10 -> %18, Ar11 -> %17, Ar17 -> %19,
Vbase -> %7
```

FIG. 35

```
        or      %0, %16          ~4800
        st      %16, %7+i        ~4802
          or      %0, 1, %17
        ld      %7+x, %18
        ld      %7+n, %19
L1 :    add     %7, %16, %20
        ld      %20+a, %21
        sub     %21, %0, %15
        ble     L2
        sub     %21, %18, %15
        bge     L5
        st      %17, %20+b
        ba      L5
L2 :    st      %17, %20+c
        add     %16, 1, %16
        sub     %19, %16, %15
        ble     L1
```

FIG. 36

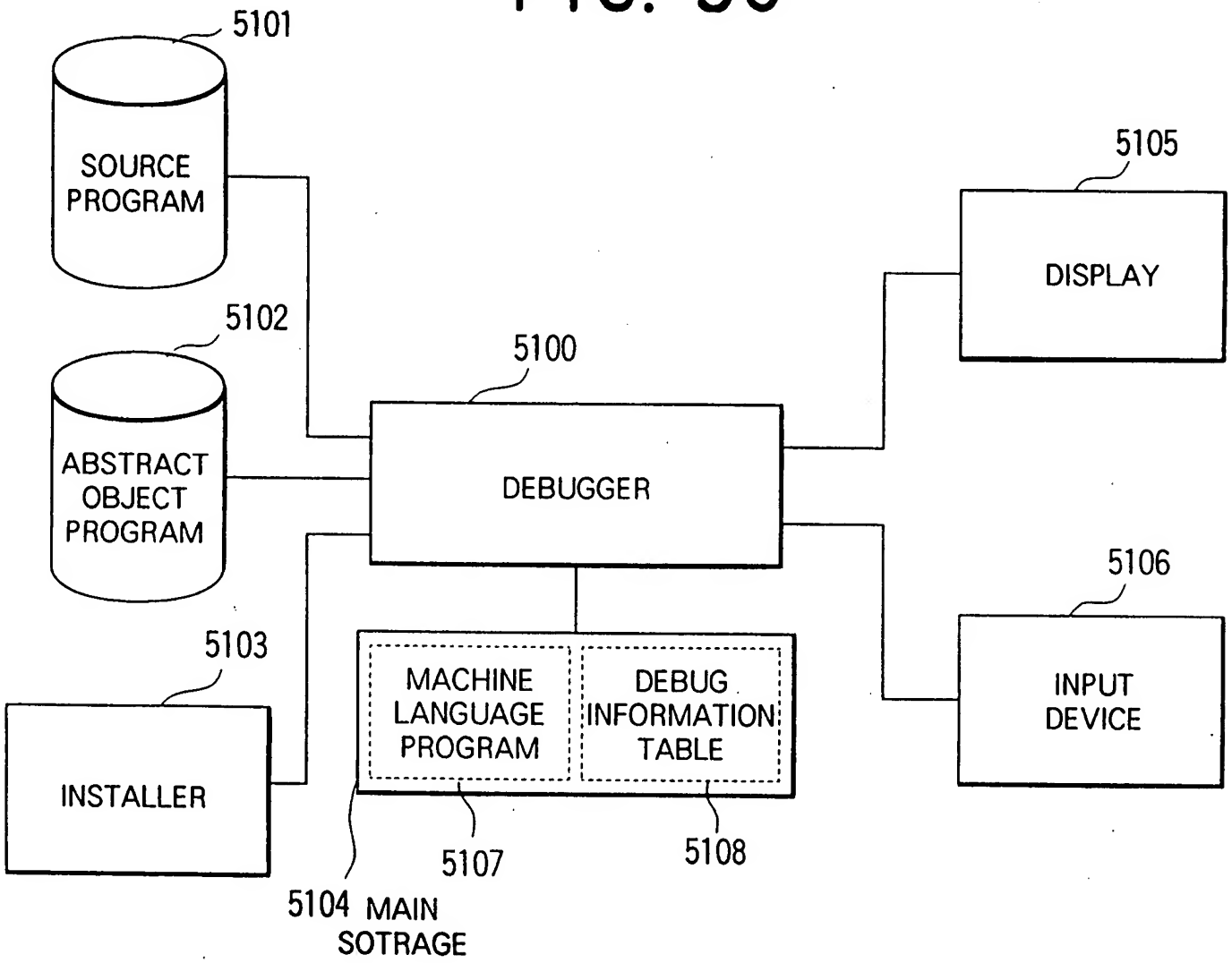

5101
SOURCE PROGRAM
5102
ABSTRACT OBJECT PROGRAM
5103
INSTALLER
5100
DEBUGGER
MACHINE LANGUAGE PROGRAM
DEBUG INFORMATION TABLE
5104 MAIN SOTRAGE
5107
5108
5105
DISPLAY
5106
INPUT DEVICE

FIG. 37

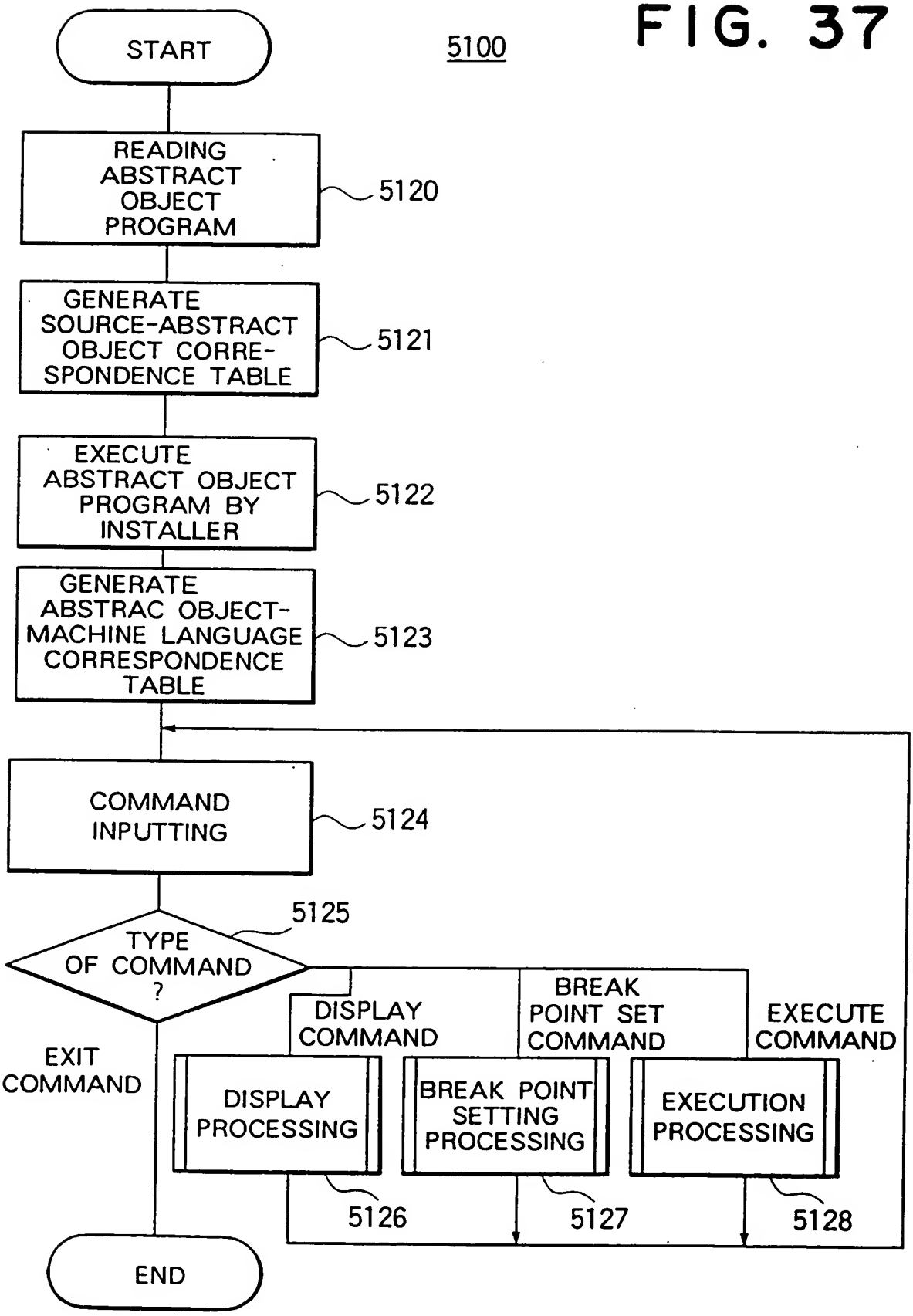

START
5100
READING ABSTRACT OBJECT PROGRAM
5120
GENERATE SOURCE-ABSTRACT OBJECT CORRE-SPONDENCE TABLE
5121
EXECUTE ABSTRACT OBJECT PROGRAM BY INSTALLER
5122
GENERATE ABSTRAC OBJECT-MACHINE LANGUAGE CORRESPONDENCE TABLE
5123
COMMAND INPUTTING
5124
5125
TYPE OF COMMAND ?
EXIT COMMAND
DISPLAY COMMAND
BREAK POINT SET COMMAND
EXECUTE COMMAND
DISPLAY PROCESSING
BREAK POINT SETTING PROCESSING
EXECUTION PROCESSING
5126
5127
5128
END

FIG. 38

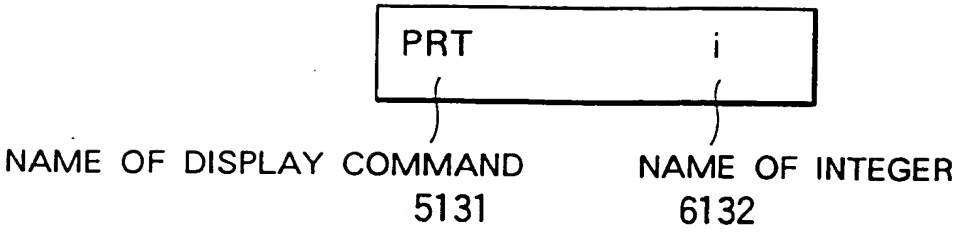

PRT
i
NAME OF DISPLAY COMMAND
5131
NAME OF INTEGER
6132

FIG. 39

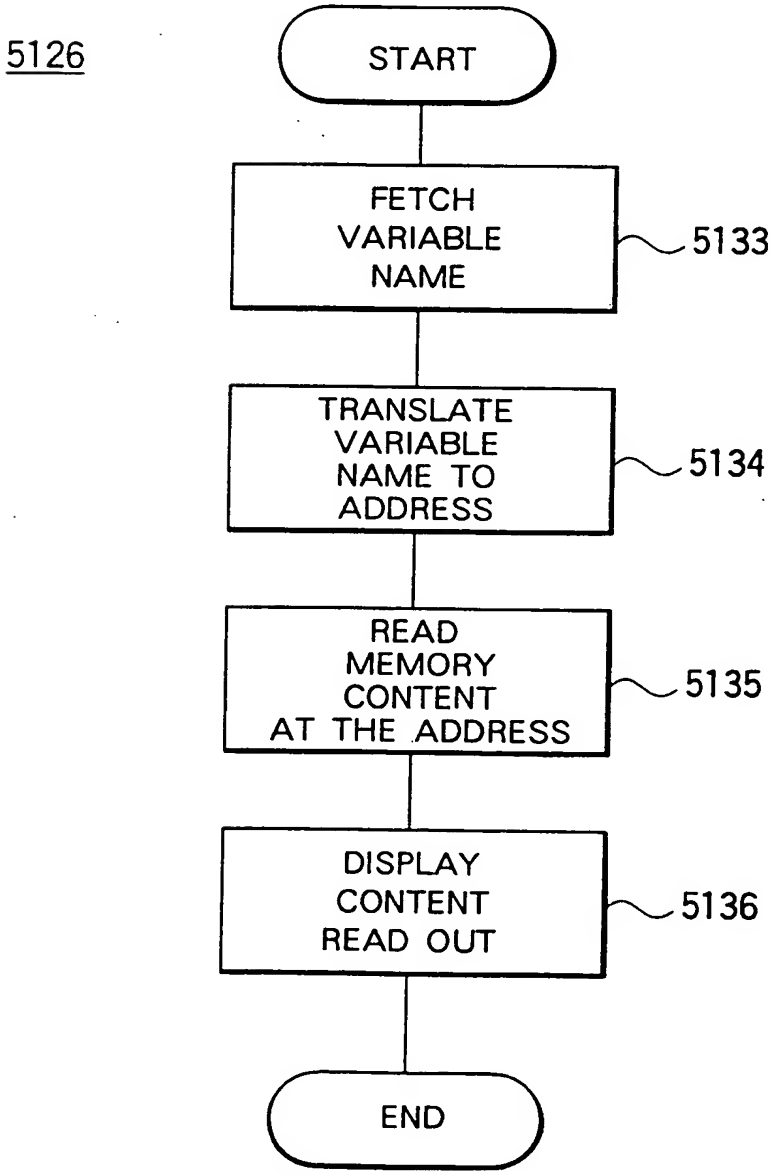

5126
START
FETCH VARIABLE NAME
5133
TRANSLATE VARIABLE NAME TO ADDRESS
5134
READ MEMORY CONTENT AT THE ADDRESS
5135
DISPLAY CONTENT READ OUT
5136
END

FIG. 40

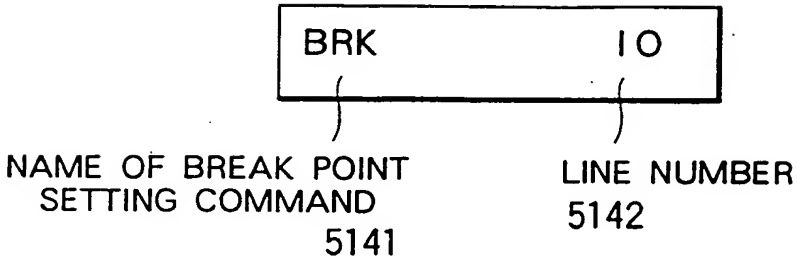

BRK
10
NAME OF BREAK POINT SETTING COMMAND
5141
LINE NUMBER
5142

FIG. 41

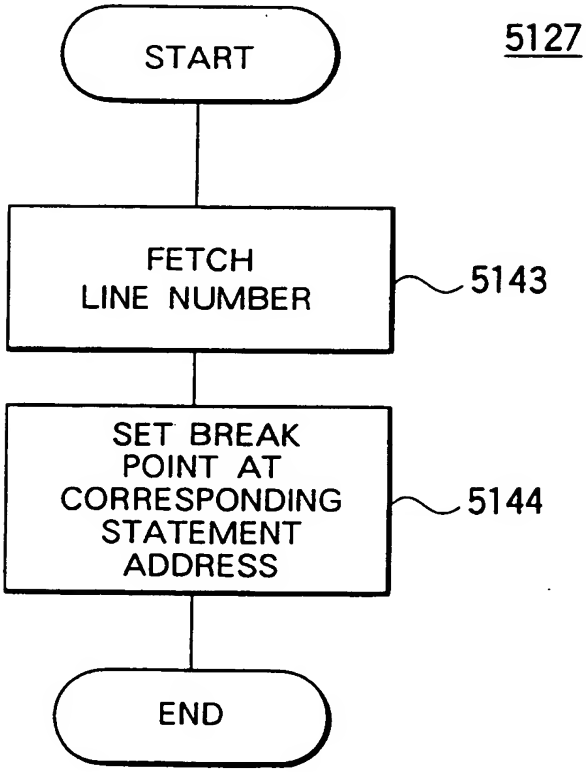

START
5127
FETCH LINE NUMBER
5143
SET BREAK POINT AT CORRESPONDING STATEMENT ADDRESS
5144
END

FIG. 42

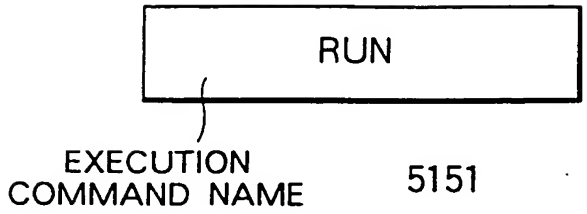

RUN
EXECUTION COMMAND NAME
5151

FIG. 43

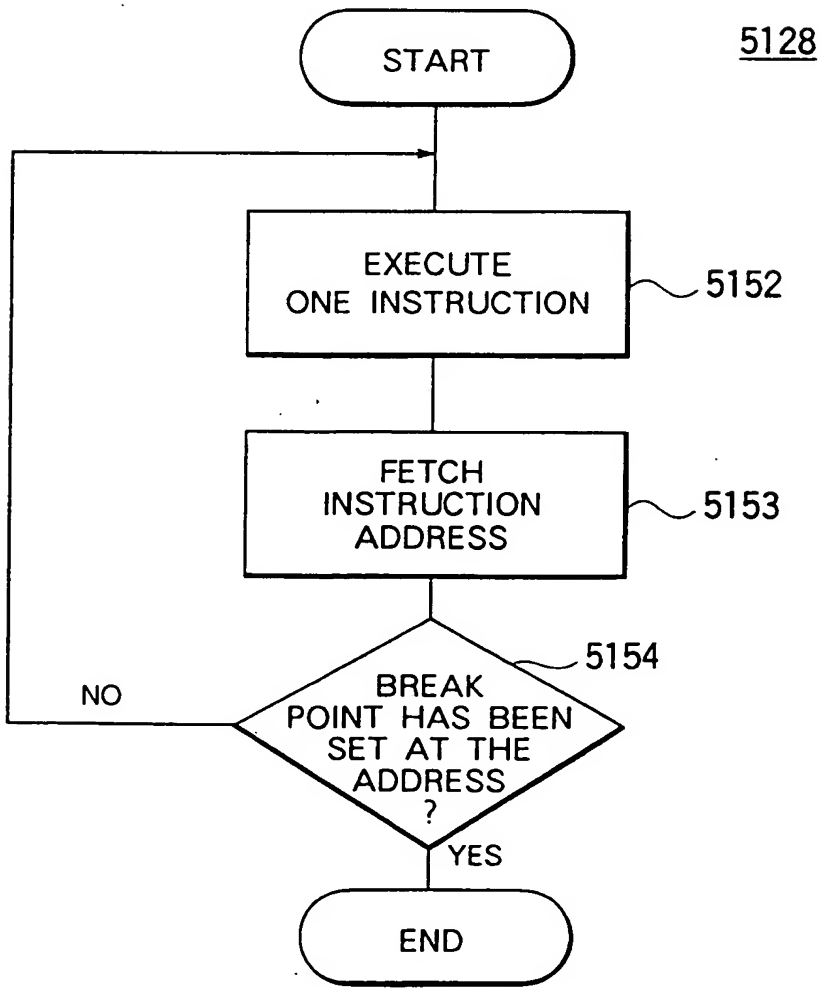

START
5128
EXECUTE ONE INSTRUCTION
5152
FETCH INSTRUCTION ADDRESS
5153
BREAK POINT HAS BEEN SET AT THE ADDRESS ?
5154
NO
YES
END

FIG. 44

5160 STATEMENT NUMBER	5161 DESCRIPTION OF SOURCE PROGRAM
1	int x ;
2	int a [10], b [10], c [10] ;
3	func ( )
4	{ int i, n ;
...	....
10	for (i=0 ; i<n ; tti) ;
11	if (a [i] >0 ;
12	if (a [i] <x)
13	b [i] =1 ;
	}
14	else c [i] =1 ;
...	....

FIG. 45A STATEMENT INFORMATION TABLE

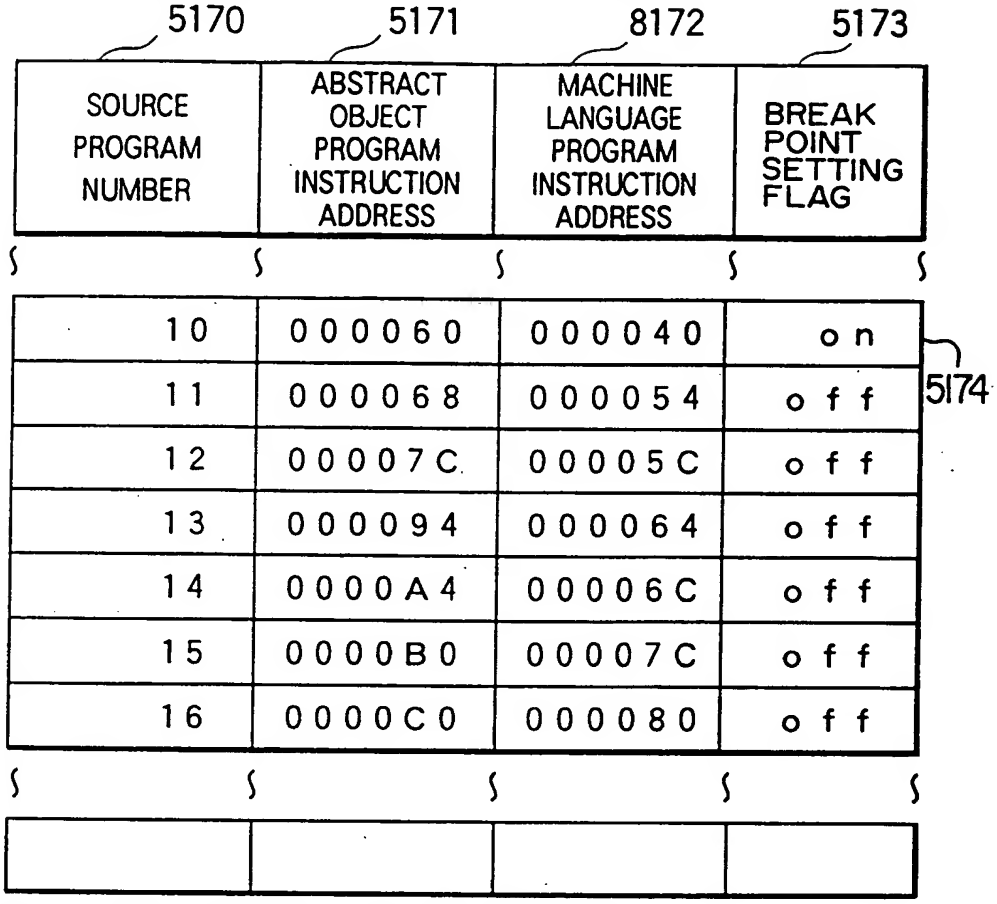


FIG. 45B VARIABLE INFORMATION TABLE

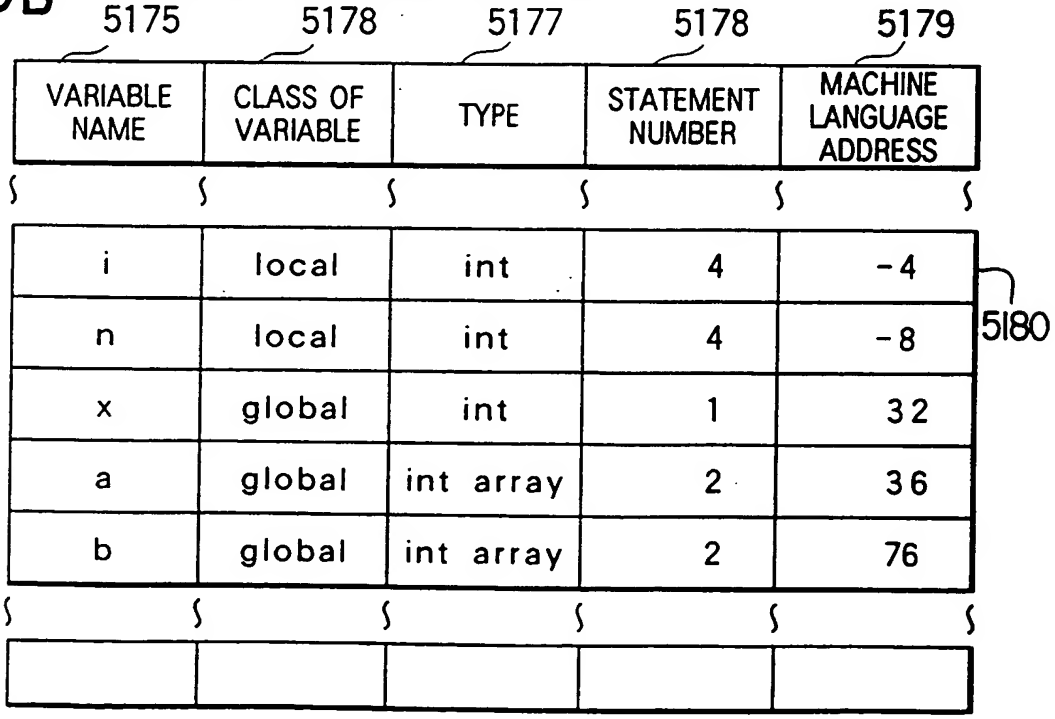


FIG. 46

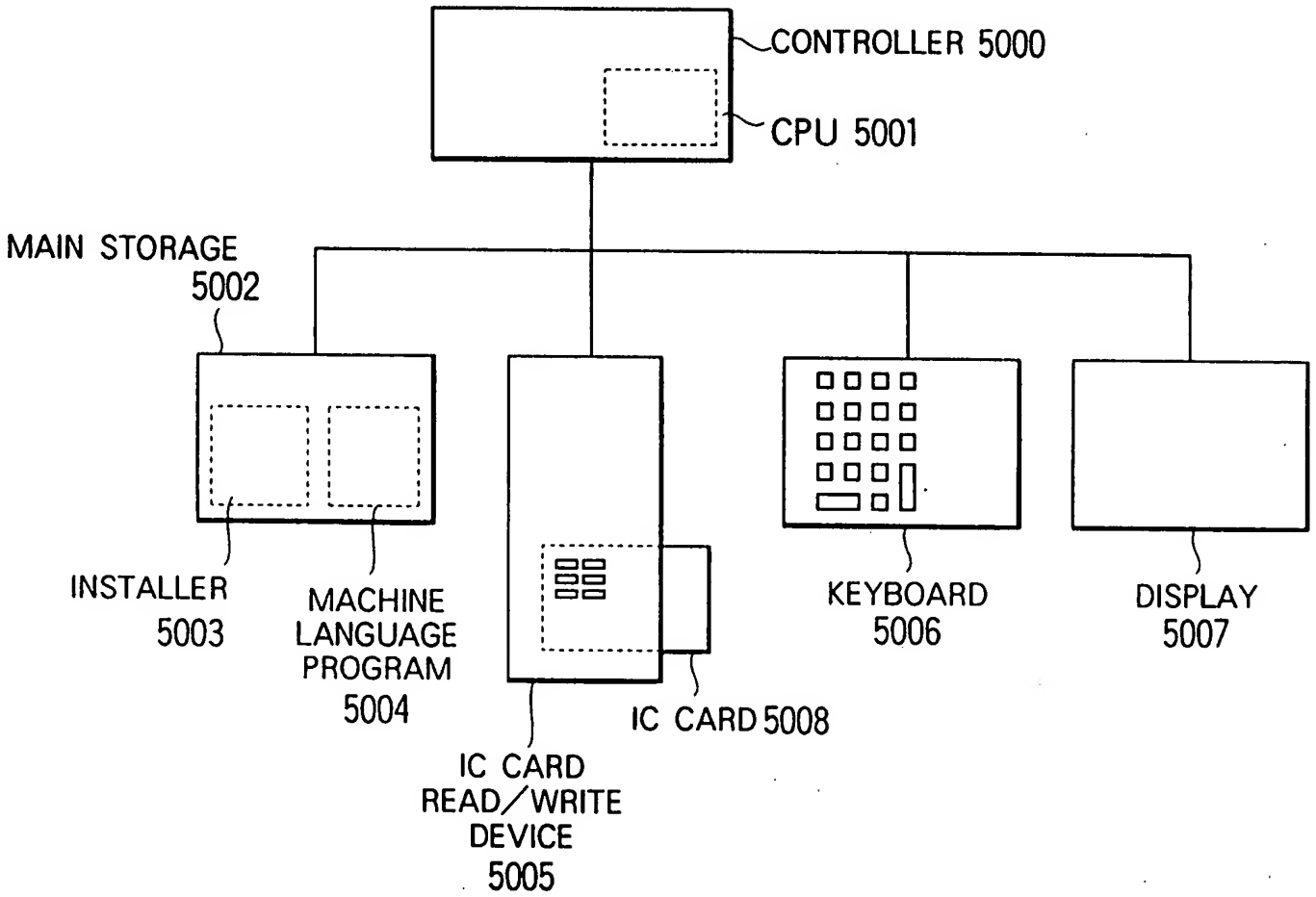

CONTROLLER 5000
CPU 5001
MAIN STORAGE
5002
INSTALLER
5003
MACHINE
LANGUAGE
PROGRAM
5004
IC CARD
READ/WRITE
DEVICE
5005
IC CARD 5008
KEYBOARD
5006
DISPLAY
5007

FIG. 47

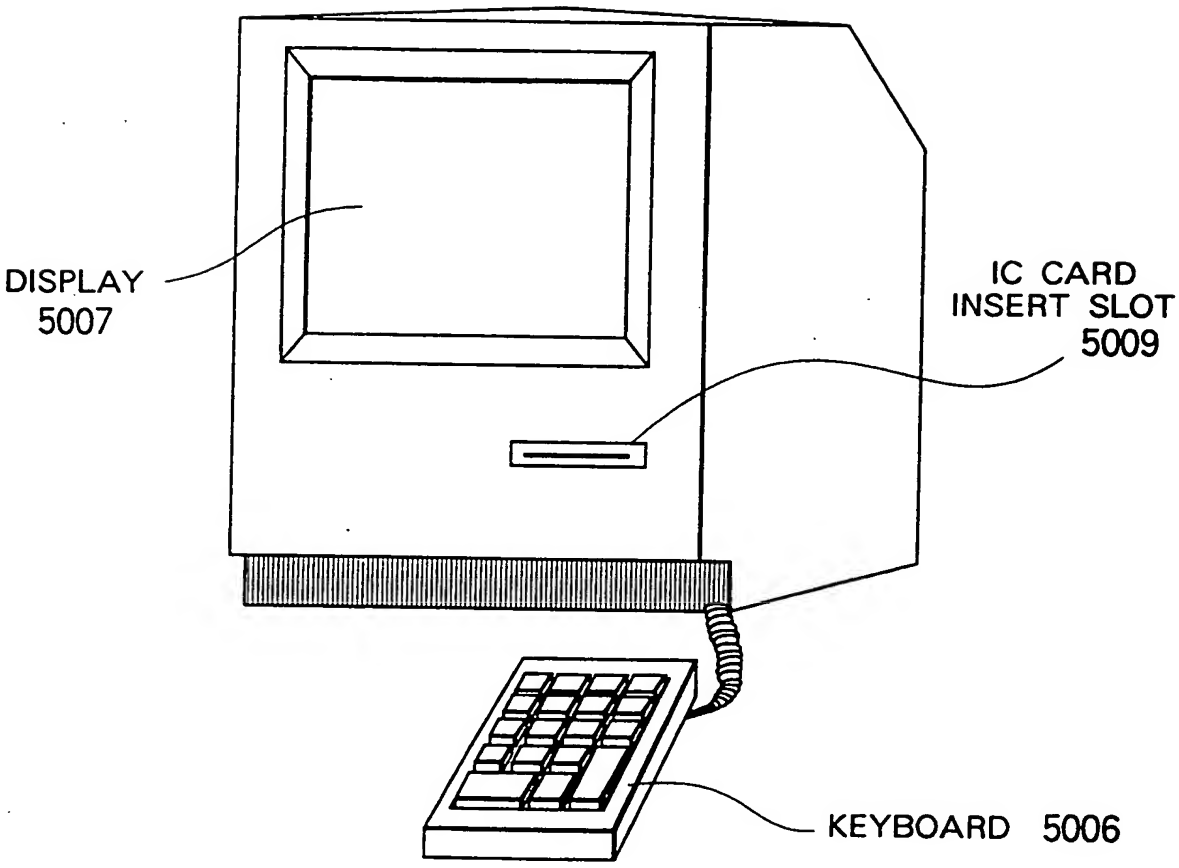


FIG. 48

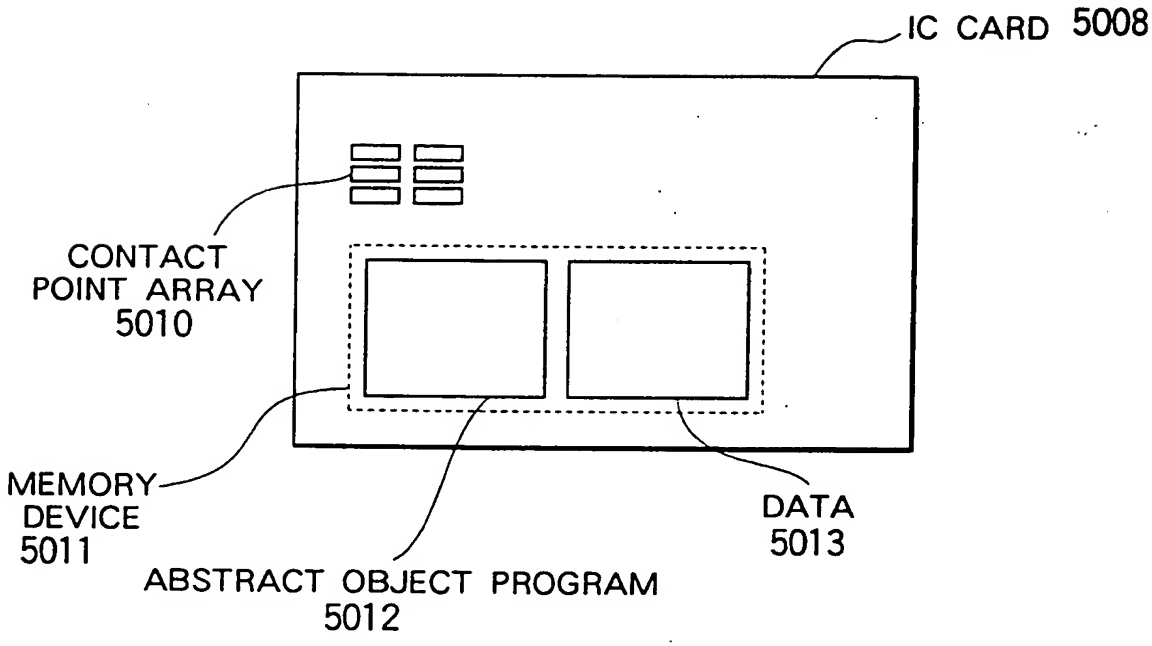
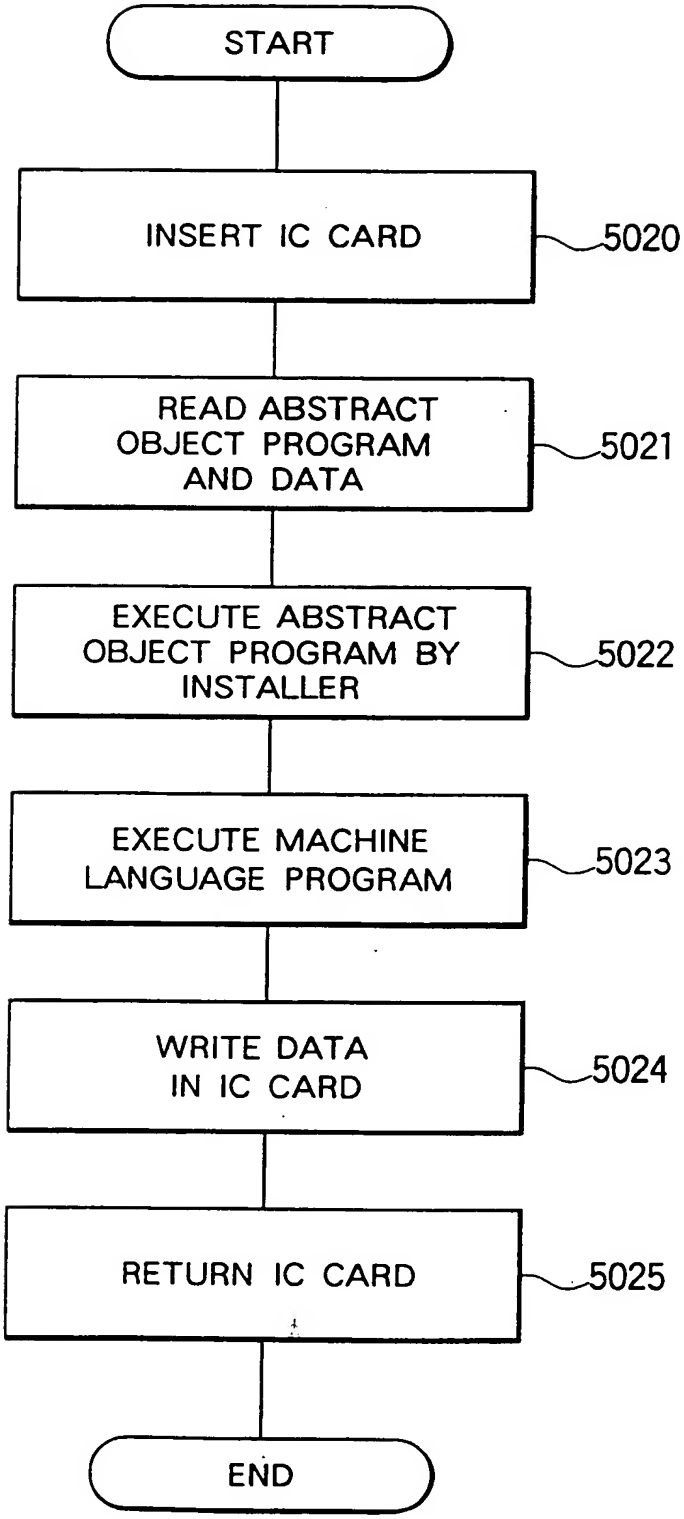

IC CARD 5008
CONTACT POINT ARRAY 5010
MEMORY DEVICE 5011
ABSTRACT OBJECT PROGRAM 5012
DATA 5013

FIG. 49


START
INSERT IC CARD
5020
READ ABSTRACT OBJECT PROGRAM AND DATA
5021
EXECUTE ABSTRACT OBJECT PROGRAM BY INSTALLER
5022
EXECUTE MACHINE LANGUAGE PROGRAM
5023
WRITE DATA IN IC CARD
5024
RETURN IC CARD
5025
END

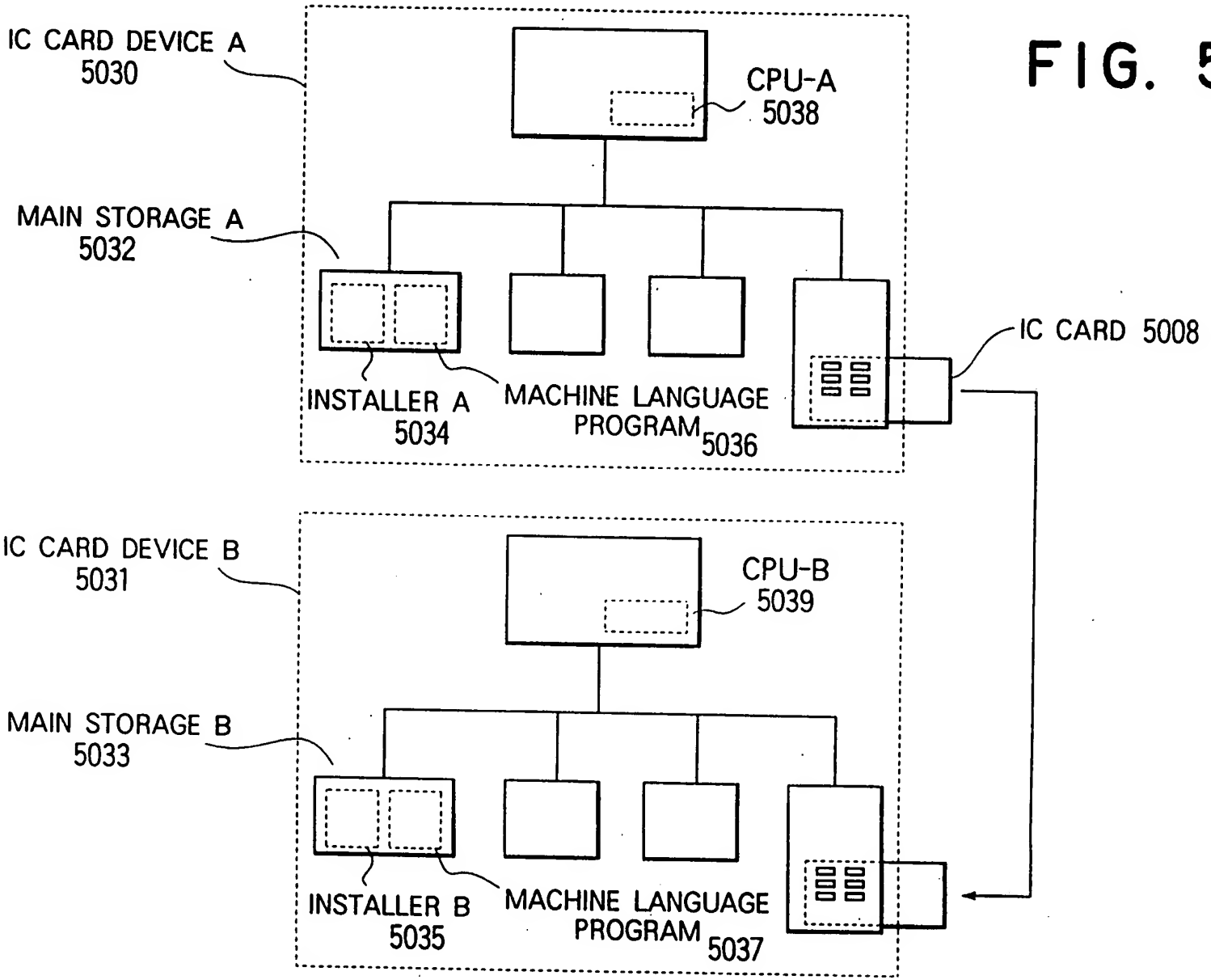

IC CARD DEVICE A
5030
CPU-A
5038
MAIN STORAGE A
5032
IC CARD 5008
INSTALLER A
5034
MACHINE LANGUAGE
PROGRAM
5036
IC CARD DEVICE B
5031
CPU-B
5039
MAIN STORAGE B
5033
INSTALLER B
5035
MACHINE LANGUAGE
PROGRAM
5037

FIG. 50

FIG. 51

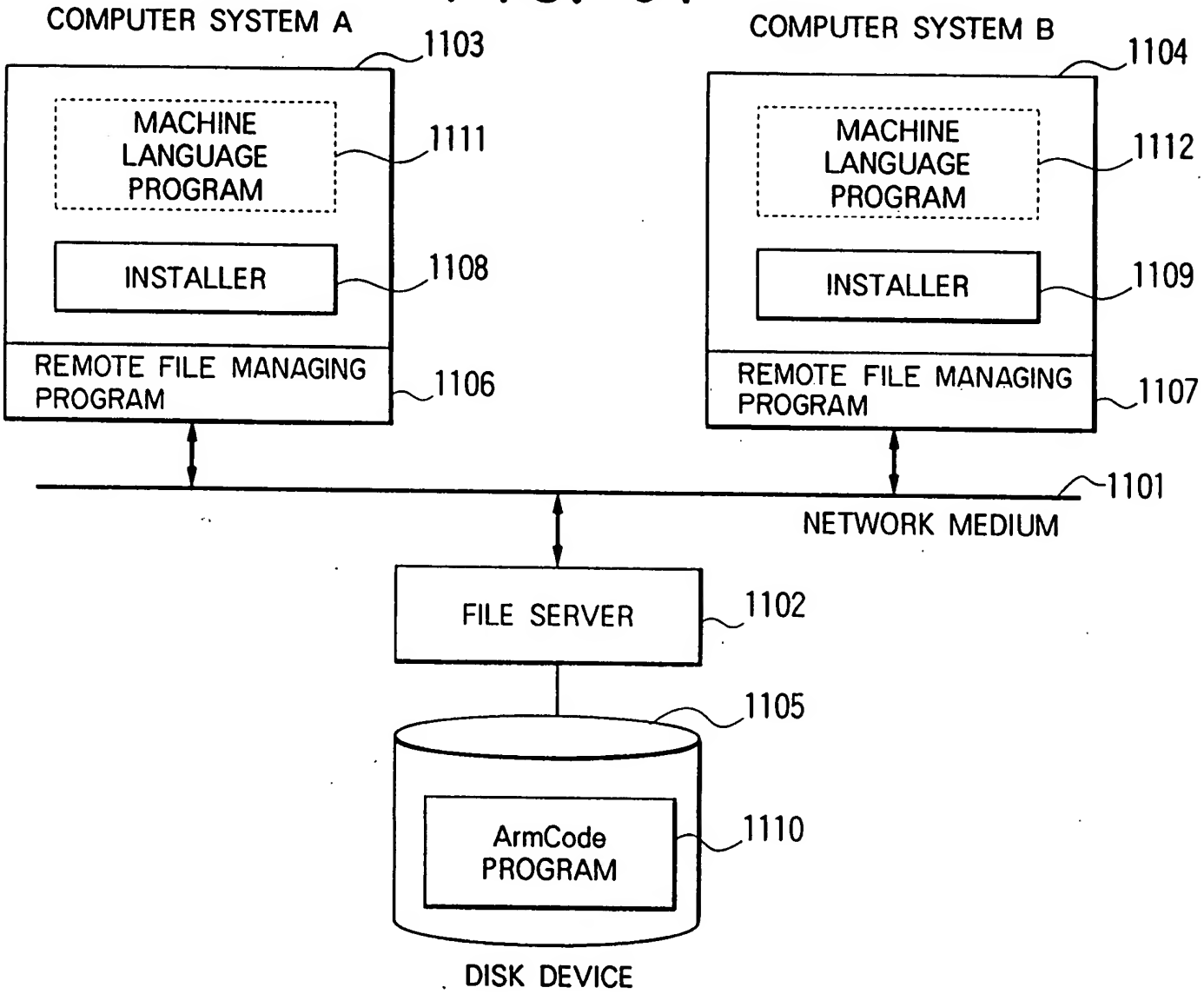
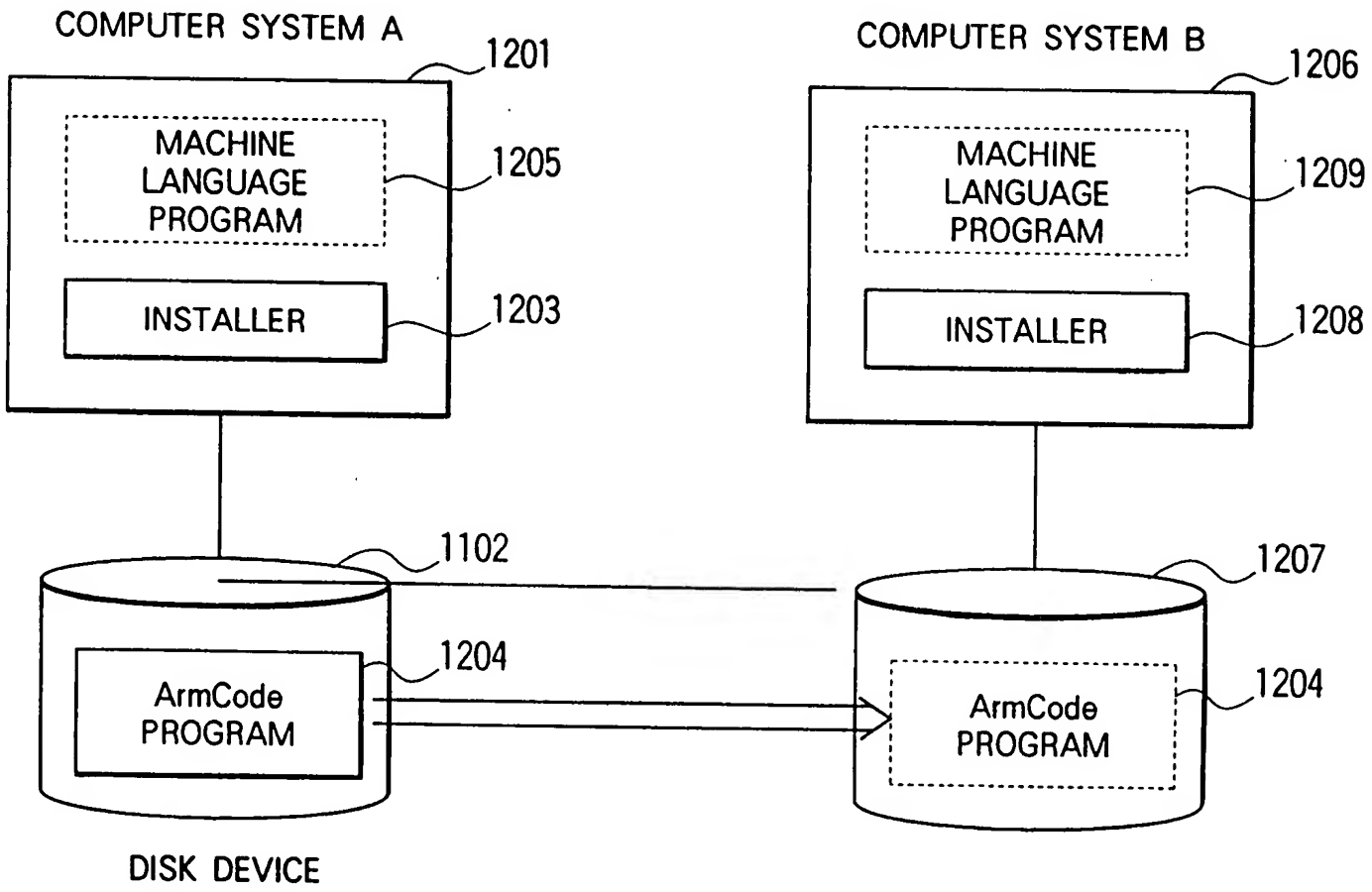

COMPUTER SYSTEM A
1103
MACHINE LANGUAGE PROGRAM
1111
INSTALLER
1108
REMOTE FILE MANAGING PROGRAM
1106
COMPUTER SYSTEM B
1104
MACHINE LANGUAGE PROGRAM
1112
INSTALLER
1109
REMOTE FILE MANAGING PROGRAM
1107
1101
NETWORK MEDIUM
FILE SERVER
1102
1105
ArmCode PROGRAM
1110
DISK DEVICE

FIG. 52


COMPUTER SYSTEM A
1201
MACHINE LANGUAGE PROGRAM
1205
INSTALLER
1203
COMPUTER SYSTEM B
1206
MACHINE LANGUAGE PROGRAM
1209
INSTALLER
1208
1102
1204
ArmCode PROGRAM
1207
1204
ArmCode PROGRAM
DISK DEVICE